21世纪高等学校规划教材 | 计算机应用



Android 应用开发教程

张冬玲 杨宁 编著

清华大学出版社

21世纪高等学校规划教材·计算机应用

Android应用开发教程

张冬玲　杨　宁　编著

清华大学出版社
北　京

内 容 简 介

本书以 Android 2.3.3 为系统平台，结合手机博客等常见的应用，较全面地介绍了 Android 应用项目的结构、控制机制，Android 的常用控件及事件处理，数据存储，多媒体应用，后台处理，网络与定位，手机基本功能等应用开发。最后通过一个综合的项目实例，给学习者一个纵观全局的实战体验。

全书共 12 章，4 大部分。第 1 部分介绍 Android 概述及基本概念；第 2 部分介绍 Android 应用程序的开发入门；第 3 部分介绍 Android 的数据存储，多媒体应用开发，后台处理等开发进阶内容；第 4 部分介绍一个综合应用实例的开发设计。

本书内容全面，案例丰富，实践性强，各章节讲述透彻，注重知识的来龙去脉，案例解析清晰。不仅可作为本科院校、大专院校、手机应用软件培训机构的相关课程教材，而且也可以作为无线互联网应用开发设计人员的参考用书。

图书在版编目(CIP)数据

Android 应用开发教程/张冬玲等编著. --北京：清华大学出版社，2013.4
21 世纪高等学校规划教材 • 计算机应用
ISBN 978-7-302-30683-2

Ⅰ. ①A…　Ⅱ. ①张…　Ⅲ. ①移动终端－应用程序－程序设计－高等学校－教材　Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2012)第 278429 号

责任编辑：高买花　薛　阳
封面设计：
责任校对：时翠兰
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　**印　张**：29.5　**字　　数**：754 千字
版　　次：2013 年 4 月第 1 版　**印　　次**：2013 年 4 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：047783-01

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施"高等学校本科教学质量与教学改革工程"（简称"质量工程"），通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部"质量工程"的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程（群）加以规划、整理和总结，更新教学内容、改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合"质量工程"的实施，满足各高校教学质量和教学改革的需要。

为了深入贯彻落实教育部《关于加强高等学校本科教学工作，提高教学质量的若干意见》精神，紧密配合教育部已经启动的"高等学校教学质量与教学改革工程精品课程建设工作"，在有关专家、教授的倡议和有关部门的大力支持下，我们组织并成立了"清华大学出版社教材编审委员会"（以下简称"编委会"），旨在配合教育部制定精品课程教材的出版规划，讨论并实施精品课程教材的编写与出版工作。"编委会"成员皆来自全国各类高等学校教学与科研第一线的骨干教师，其中许多教师为各校相关院、系主管教学的院长或系主任。

按照教育部的要求，"编委会"一致认为，精品课程的建设工作从开始就要坚持高标准、严要求，处于一个比较高的起点上。精品课程教材应该能够反映各高校教学改革与课程建设的需要，要有特色风格、有创新性（新体系、新内容、新手段、新思路，教材的内容体系有较高的科学创新、技术创新和理念创新的含量）、先进性（对原有的学科体系有实质性的改革和发展，顺应并符合 21 世纪教学发展的规律，代表并引领课程发展的趋势和方向）、示范性（教材所体现的课程体系具有较广泛的辐射性和示范性）和一定的前瞻性。教材由个人申报或各校推荐（通过所在高校的"编委会"成员推荐），经"编委会"认真评审，最后由清华大学出版社审定出版。

目前，针对计算机类和电子信息类相关专业成立了两个"编委会"，即"清华大学出版社计

算机教材编审委员会”和“清华大学出版社电子信息教材编审委员会”。推出的特色精品教材包括：

（1）21世纪高等学校规划教材·计算机应用——高等学校各类专业，特别是非计算机专业的计算机应用类教材。

（2）21世纪高等学校规划教材·计算机科学与技术——高等学校计算机相关专业的教材。

（3）21世纪高等学校规划教材·电子信息——高等学校电子信息相关专业的教材。

（4）21世纪高等学校规划教材·软件工程——高等学校软件工程相关专业的教材。

（5）21世纪高等学校规划教材·信息管理与信息系统。

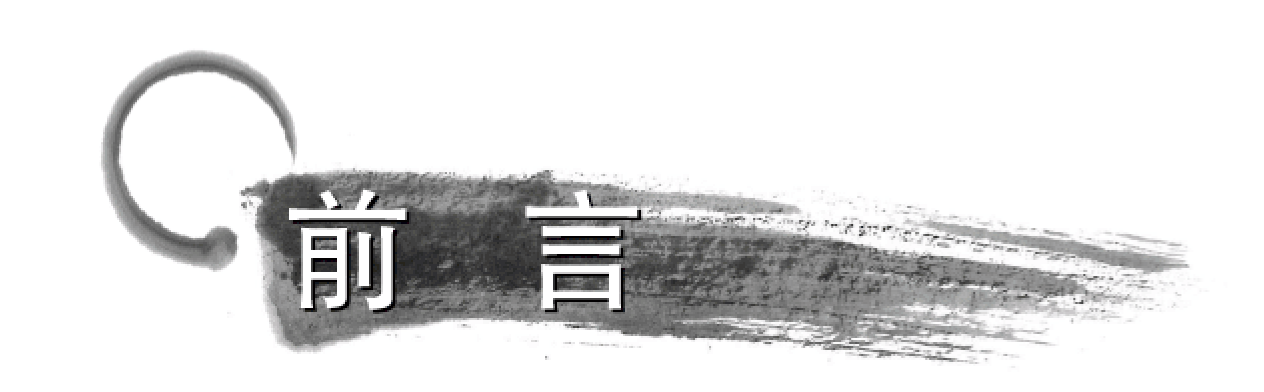
（6）21世纪高等学校规划教材·财经管理与应用。

（7）21世纪高等学校规划教材·电子商务。

（8）21世纪高等学校规划教材·物联网。

清华大学出版社经过三十多年的努力，在教材尤其是计算机和电子信息类专业教材出版方面树立了权威品牌，为我国的高等教育事业做出了重要贡献。清华版教材形成了技术准确、内容严谨的独特风格，这种风格将延续并反映在特色精品教材的建设中。

清华大学出版社教材编审委员会
联系人：魏江江
E-mail：weijj@tup.tsinghua.edu.cn



Android 是一个优秀的开源手机平台。本书以 Android 2.3.3 为系统平台，使用 Eclipse 为开发工具，介绍在 Android 平台上进行应用开发的知识和技术。本书吸收了 Android 开发设计类书籍的优点，总结了一些培训机构的教学经验，从教学的角度全面介绍了 Android 应用程序的开发设计，深浅适宜，实例丰富，不仅可作为本科院校、大专院校、高职高专和计算机培训机构的相关课程的教材，而且也可作为 Android 系统开发设计人员的开发参考书。

全书共 12 章，内容分为以下 4 部分。

第 1 部分是 Android 概述及基本概念，由第 1，2 章描述，介绍 Android 开发起步，Android 应用程序的构成。

第 2 部分是开发入门，由第 3～6 章描述，介绍 Android 应用程序的控制机制，常用基本控件，高级控件及事件处理应用，菜单与对话框等内容。

第 3 部分是开发进阶，由第 7～11 章描述，介绍 Android 的数据存储，多媒体应用开发，后台处理，网络与位置地图，手机基本功能开发等内容。

第 4 部分是综合应用实例开发，由第 12 章描述，介绍应用项目实例的开发设计。

Android 课程内容十分丰富，实践性强，教学课时建议不低于 90 学时，并且需要保证充足的实践课时数，建议实践课课时不低于 48 学时。

本书凝聚了作者多年的教学与手机开发经验，讲解深入透彻，论述通俗易懂，注重知识的系统性，案例解析清晰透彻。凡具备编程基础的人员都可以通过本书的学习掌握 Android 的应用编程。

本书的主要章节由张冬玲编写。杨宁完成第 12 章，包括应用实例的系统功能设计，服务器端、手机客户端应用项目的编程，以及通信协议的编写等内容。刘雄负责本教程的部分案例的设计。芦晓燚完成各章的练习题编写。全书由张冬玲统稿与定稿。

由于作者水平有限，书中难免会有疏漏与不足之处，敬请各位专家与读者指正。

编者　张冬玲

于中山大学新华学院(广州)

2012 年 8 月 26 日

目 录

第1章 Android开发起步

1.1 Android 移动开发平台概述

1.1.1 认识 Android

Android 系统是由 Google 公司推出的手机操作系统。它的标志是一个名叫 Android 的小机器人，它来源于法国作家利尔·亚当在 1886 年发表的科幻小说《未来夏娃》中，作者为小说里的机器人起名为 Android。最开始的时候，研发 Android 系统是由一个名为 Android 的公司进行的，2005 年 Google 公司收购了这个仅成立 22 个月的高科技企业，由 Google 公司接手研发 Android 系统，Android 系统的创始人，Android 公司的 CEO 安迪·鲁宾，成为 Google 公司的工程部副总裁，继续负责 Android 项目的研发工作。

2007 年 11 月 5 日，Google 公司正式向外界展示了 Android 的操作系统，并且宣布将建立一个全球性的联盟组织，该组织由摩托罗拉、索尼爱立信 Sony Ericsson、韩国三星电子、韩国 LG 电子、台湾宏达国际电子 HTC、中国移动 China Mobile、意法半导体 ST、英特尔 Intel、Skype(eBay)等 34 家手机制造商、软件开发商、电信运营商以及芯片制造商共同组成。这一联盟将支持 Google 发布的手机操作系统以及应用软件，将共同开发 Android 系统的开放源代码。2008 年 9 月，Google 正式发布了 Android 1.0 系统，这也是 Android 系统最早的版本。从此，Google 开启了一个新的手机系统辉煌时代。

目前手机市场上，随着 Android 平台的发展以及不断完善，越来越多的手机厂商开始选择 Android 系统作为其主要发展方向，Android 系统在全球手机市场上已经占据市场份额第一的位置许久。虽然随着近期 WP7 的推出，手机智能系统的竞争愈加激烈。但就目前来说，Android 手机的统治地位还是无法改变的。

1.1.2 Android 的发展历史

2007 年 11 月，Google 公司宣布其基于 Linux 平台的开源手机操作系统的项目代号为“Android”。

2008 年 3 月，Android SDK 发布，代号为 m5-rc15；同年 8 月，Android 0.9 SDK beta 版本发布，代号为 m5-0.9；同年 9 月 23 日，美国运营商 T-Mobile 在纽约正式发布第一款 Android 手机 T-Mobile G1，它成为 Android 1.0 代表机型。该款手机由台湾宏达电子(HTC)代工制造，是世界上第一部使用 Android 操作系统的手机，支持 WCDMA 网络，并支持 WiFi；当天，Android 1.0 SDK 发布。

2009年2月,Android 1.1 SDK发布;同年5月,Android 1.5 SDK发布,其代表机型为HTC G2,G1和G2逐渐被市场接受。从Android 1.5版本开始,Google开始将Android的版本以甜品的名字命名,Android 1.5命名为Cupcake(纸杯蛋糕)。该系统与Android 1.0相比有了很大的改进。如:摄/播放影片,并支持上传到Youtube;最新的采用WebKit技术的浏览器,支持复制/粘贴和页面中搜索;提供屏幕虚拟键盘;应用程序自动随着手机旋转;短信、Gmail、日历,浏览器的用户接口大幅改进,如Gmail可以批量删除邮件;来电照片显示等。

2009年9月,Android 1.6 SDK发布,其代表机型为HTC Hero G3,HTC Hero G3成为最受欢迎的机型。Google为Android 1.6命名为代号为Donut(甜甜圈)。Android 1.6改进如下:重新设计的Android Market手势,支持CDMA网络,文字转语音系统(Text-to-Speech),快速搜索框,全新的拍照接口,查看应用程序耗电,支持虚拟私人网络(VPN),支持更多的屏幕分辨率,支持OpenCore 2媒体引擎,新增面向视觉或听觉困难人群的易用性插件等。

2009年10月,发布了Android 2.0操作系统,2010年1月,发布Android 2.1,并推出Google旗下第一款自主品牌手机Nexus One,该机成为Android 2.1的代表机型,由HTC代工生产。此时,众多厂商加盟支持,机型越来越多,市场反应和占有率越来越高。Google将Android 2.0至Android 2.1系统的版本统称为Eclair(松饼),新系统与旧系统相比进行了较大的改进。Android 2.0～2.1改进如下:优化硬件速度,改良的用户界面,新的浏览器的用户接口和支持HTML5,改进Google Maps 3.1.2,支持内置相机闪光灯,支持数码变焦,改进的虚拟键盘,支持蓝牙2.1,支持动态桌面的设计等。

2010年5月,Android 2.2 SDK发布,其代表机型为DHD和GALAXY S。Google为其命名为Froyo(冻酸奶)。Android 2.2操作系统在当时受到了广泛的关注,根据美国NDP集团调查显示,在当时Android系统已占据了美国移动系统市场28%的份额,在全球占据了17%的市场份额。到2010年9月,Android系统的应用数量已经超过了9万个,Google公布每日销售的Android系统设备的新用户数量达到20万,Android系统取得了巨大的成功。Android 2.2改进如下:整体性能大幅度的提升,3G网络共享功能,支持Flash,App2sd功能,全新的软件商店,更多的Web应用API的开发。

2010年10月,Google宣布Android系统达到了第一个里程碑,即电子市场上获得官方数字认证的Android应用数量已经达到了10万个,Android系统的应用增长非常迅速。2010年12月,Google正式发布了Android 2.3操作系统Gingerbread(姜饼),其代表机型是三星的GALAXYSⅡ和HTC Sensation。Android 2.3改进如下:增加了新的垃圾回收和优化处理事件,原生代码可直接存取输入和感应器事件,EGL/OpenGL ES,OpenSL ES,新的管理窗口和生命周期的框架,支持VP8和WebM视频格式,提供AAC和AMR宽频编码,提供了新的音频效果器,支持前置摄像头、SIP/VOIP和NFC(近场通信),简化界面,速度提升,一键文字选择和复制/粘贴,改进的电源管理系统,新的应用管理方式等。

2011年1月,第一款采用Android 3平台的平板计算机问世。同年2月,Android 3 SDK发布。Android 3.0是专门针对平板计算机进行优化的操作系统。Android进入平板计算机时代。

2011年7月,Google称每日的Android设备新用户数量达到了55万部,而Android系统设备的用户总数达到了1.35亿,Android系统已经成为智能手机领域占有量最高的系统。2011年9月,Google发布了全新的Android 4.0操作系统,Android 4.0的代表机型就是NEXUS Prime和Droid Razr。这款系统被命名为Ice Cream Sandwich(冰激凌三明治)。这款全新的Android系统结合了Android 2.3与Android 3.0的优点,支持手机设备与平板设备。Android 4.0系统拥有全新的系统解锁界面,小插件也进行了重新设计,最特别的就是系

统的任务管理器可以显示出程序的缩略图,便于用户准确快速地关闭无用的程序。

对 Android 来说,最大的特点就是具有开源性。其特点在于改变以往由少数软件大厂垄断系统软件平台的现况,让众多内容开发商和开放软件供货商来分享共同利益,极大地增进了客户使用经验。这是 Android 系统市场份额大力攀升的原因之一。

纵览 Android 发展历程,在这短短的三年多时间里,Android 系统的发展经历了翻天覆地的变化。在这高速的成长过程中,也存在一些问题。例如,由于其开源性导致衍生版本过于混乱、终端设备的硬件配置良莠不齐,使得 Android 的兼容性不断降低,给开发者开发应用程序带来了很大的不便;在 Android 3.0 发布后,开源与否就成了争议的焦点,造成了 Android 平台的分化,不利于 Android 的发展;Android 的应用程序还存在着一些安全隐患问题。对于这些问题,还需要 Google 从政策的高度对整体进行调控,从市场的角度进行引导和规范。

1.1.3 Android 主要应用

Android 从开始的手机操作系统,现在发展成为移动设备(如 PDA、MID 产品、平板计算机等)的操作系统,在这个系统平台上可以开发出通信、定位、餐饮、娱乐、商务、家电控制、行业服务等多方面的既实用又有吸引力的移动服务应用。其中,手机在移动互联网方面的应用是当前发展的主流应用。

如图 1-1 所示是一个由中国互联网络信息中心(CNNIC)于 2012 年 3 月发表的《中国移动互联网发展状况调查报告》中的统计图表。

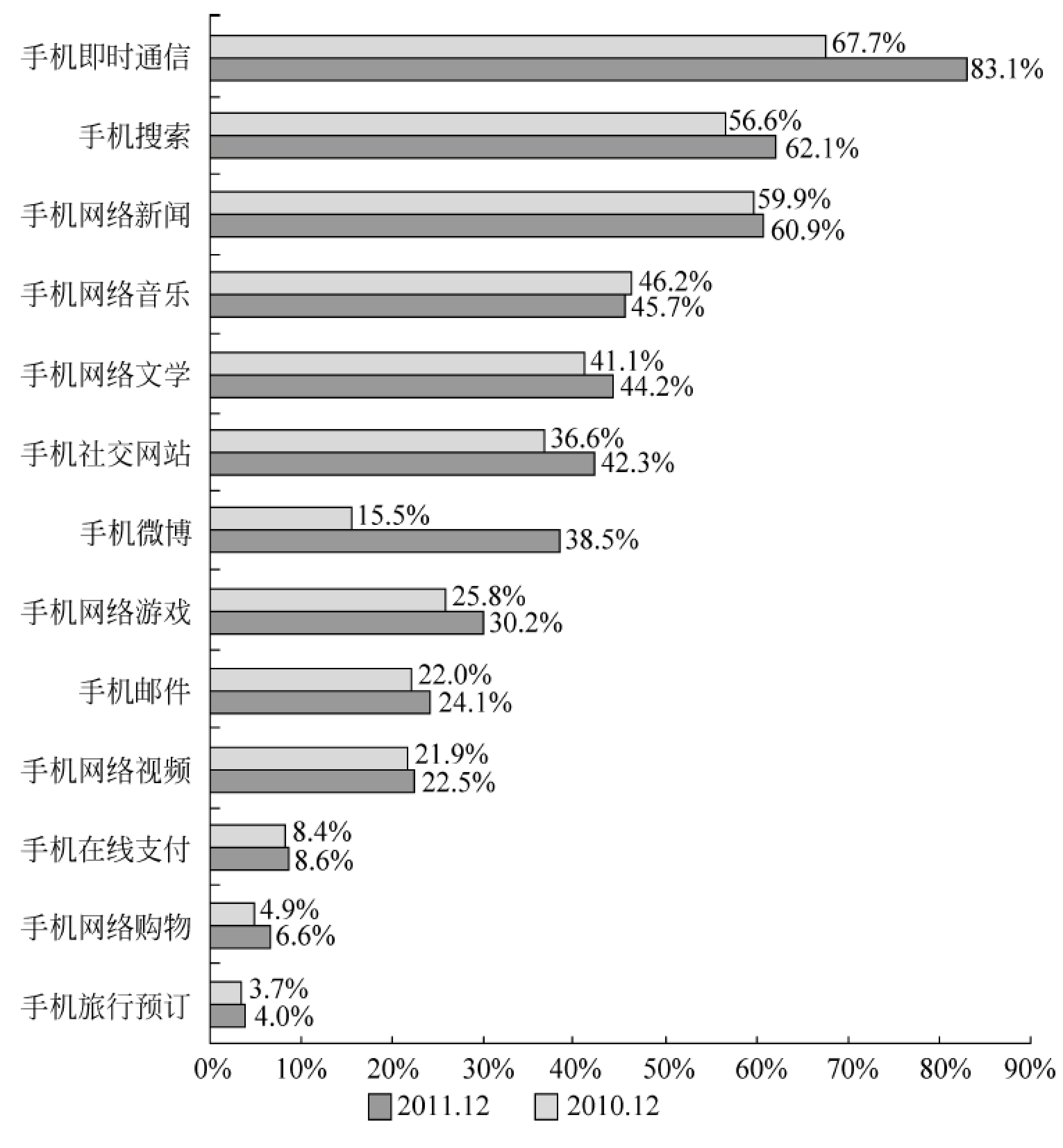


图 1-1 2010—2011 年手机网民网络应用

在移动互联网应用方面，CNNIC 预测：手机微博将是继即时通信之后，又一个吸引网民使用移动互联网的关键应用。首先，相对计算机来说，手机微博更能体现微博内容的随时随地性特点；其次，手机微博更能发挥微博应用的自媒体优势，例如即兴拍摄的图片、视频等，这反映出手机在即兴原创内容方面的能力远远高于计算机；第三，手机微博的使用并没有给用户带来不友好的阅读体验。手机搜索、手机网络新闻、手机发帖回帖、手机社交网站、手机邮件等应用是与手机即时通信一样的传统手机应用，2011 年同比使用率均有小幅度的提升。

将来主要的应用推动方向在于：手机视频娱乐类应用。手机视频作为典型的娱乐类应用，使用率变化不大。短期内手机视频业务发展主要面临以下困难：一方面，无线网络基础设施不能满足用户需求，视频应用需要消耗大量的无线网络流量，现阶段无线网络资费较高，且带宽不稳定，成为阻碍这类应用发展的瓶颈；另一方面，手机视频相应的内容较为缺乏。

电子商务类应用普遍处于发展初期，在手机网民中渗透率较低，主要原因为，一方面，大部分电子商务产品，均需要用户进行比较、咨询后才能完成购买，手机较小的屏幕使得购物体验相对较差；另一方面，大部分用户还未建立起对手机支付的信任和使用习惯。

由此可以看出，Android 应用开发将大有作为。

1.2 Android 框架简介

Android 平台采用了整合的策略思想，包括底层 Linux 操作系统、中间层的中间件和上层的 Java 应用程序。Android SDK 提供了必需的工具和进行应用开发所必需的 Java 接口 API。

1.2.1 Android 平台特点

Android 平台用户数量能在短时间内迅速激增与它所具有的特点分不开。从其架构的角度来看，Android 平台具有以下几个特点。

1. 开放性

谈到 Android 平台的特点首先就是其开放性。首先从 Android 源码上开放，使得每一个应用程序可以调用其内部的任何核心应用源码；其次是平台上开放，Android 平台不存在任何阻碍移动产业创新的专有权限制，任何联盟厂商都可以根据自己的需要自行定制基于 Android 操作系统的手机产品；再次是运营上开放，手机使用什么方式接入什么网络，已不再依赖运营商的控制，用户可以更加方便地连接网络；等等。这些显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。

2. 应用程序平等

在 Android 平台中，其内部的核心应用和第三方应用是完全平等的，用户能完全根据自己的喜好使用它们来定制手机服务系统；其应用程序框架支持组件的重用与替换，程序员可以完全平等地调用其内部核心程序或第三方应用程序。

3. 支持丰富的硬件

Android 平台支持丰富的硬件，这一点还是与 Android 平台的开放性相关，由于 Android

的开放性，众多的厂商会推出千奇百怪、功能特色各异的多种产品。

4. 众多的开发商

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，因此不会受到各种条条框框的阻挠，可想而知，会有多少新颖别致的软件诞生。但与此同时，也有些不健康的、恶意的程序和游戏出现，如何控制它们正是 Android 的难题之一。

5. 强大的 Google 应用

从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 服务。

1.2.2　Android 平台架构

Android 是一个开放的软件系统，为用户提供了丰富的移动设备开发功能。其平台架构从下至上包括 4 个层次，如图 1-2 所示。

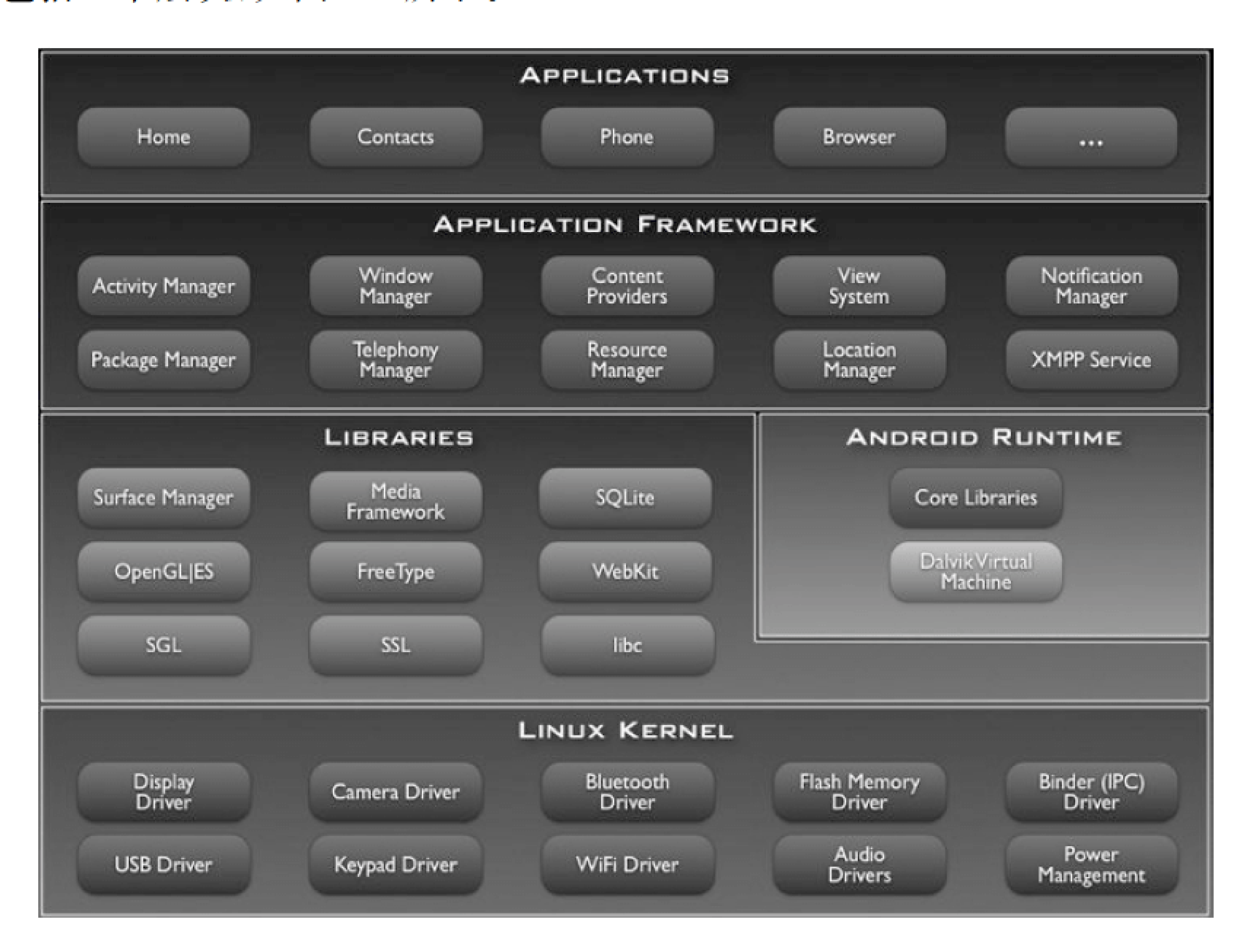


图 1-2　Android 操作系统的体系结构

从图 1-2 可以看出，Android 操作系统的体系结构由上到下依次是应用程序、应用程序框架、系统运行库和 Linux 内核，其中第三层还包括 Android 运行时的环境。下面分别来讲解各个部分。

1. 应用程序

Android 平台不仅是操作系统，也预装了一组核心应用程序，包括 E-mail 客户端、短信服务、日历日程、地图服务、浏览器、联系人和其他应用程序。所有应用程序都是 Java 编程语言编写的。

不仅如此，这些应用程序都是可以被程序员用自己编写的应用程序所替换，这点不同于其

他手机操作系统固化在系统内部的系统软件，Android 系统更加灵活和个性化。这个替换的机制实际是应用程序框架来保证的。

2. 应用程序框架

应用程序框架层是从事 Android 开发的基础，上面所提的核心应用程序就是依赖框架层次 API 开发的。应用架构设计的初衷是：简化组件重用机制；任何应用都能发布自己的功能，这些功能又可以被任何其他应用使用(当然要受来自框架的强制安全规范的约束)。和重用机制相同，框架允许组件的更换。在这个应用程序框架中，程序员可以直接使用其提供的组件来进行快速的应用程序开发，也可以通过继承而实现个性化的拓展。

所有应用框架都是一组服务和系统，一般包含以下几部分。

(1) View System(视图系统)。一套丰富且可扩展的视图组件，可以用来构建应用程序，它包括列表(lists)、网格(grids)、文本框(text boxes)、按钮(buttons)以及嵌入的网络浏览器等。

(2) Content Providers(内容提供器)。使一个应用可以访问另外一个应用的数据，或者使一个应用内部可以共享自身数据。例如手机中的联系人信息。

(3) Resource Manager(资源管理器)。提供对非编码资源的访问通道。例如本地化字符串、图片和布局文件等。

(4) Notification Manager(通知管理器)。将应用的消息显示在状态栏中，给用户以警报或通知。

(5) Activity Manager(行动管理器)。负责管理应用的生命周期，提供常用导航回退支持。

(6) Window Manager(窗口管理器)。管理所有的窗口程序。

(7) Package Manager(包管理器)。Android 系统内的程序管理。

(8) Telephony Manager(电话管理器)。管理所有的移动设备功能。

3. 系统运行库

从图 1-2 中可以看出，系统运行库层可以分成两部分，分别是系统库和 Android 运行时。

1) 系统库

Android 包含一套 C/C++库，Android 系统的各个组件都在使用，这些功能是通过 Android 应用框架暴露给开发人员的。系统库是应用程序框架的支撑，是连接应用程序框架层与 Linux 内核层的重要纽带。其主要核心库包括以下几部分。

(1) Libc(系统 C 库)。一个从 BSD 继承来的标准 C 系统函数库，专门为基于 Embedded Linux 的设备定制。

(2) Media Framework(多媒体库)。Android 系统多媒体库，基于 PacketVideo OpenCORE。该库支持录放，并且可以录制许多流行的音频视频格式，还有静态映像文件，包括 MPEG4、H.264、MP3、AAC、JPG、PNG 等。

(3) Surface Manager。主要负责管理针对显示系统的访问，并且为多个应用程序提供 2D 和 3D 图层的无缝融合。

(4) Webkit 浏览器。一个最新的 Web 浏览器引擎，用来支持 Android 浏览器和一个可嵌入的 Web 视图。

(5) SGL。一个内置的2D图形引擎。

(6) SSL。位于TCP/IP与各种应用层协议之间,为数据通信提供支持。

(7) OpenGL ES。3D效果的支持。基于OpenGL ES 1.0 APIs实现;该库可以使用硬件3D加速(如果可用)或者使用高度优化的3D软加速。

(8) FreeType。提供位图(bitmap)和向量(vector)的字体描述与显示。

(9) SQLite。一个对于所有应用程序可用、功能强劲的轻型关系型数据库引擎。

2) Android 运行时

Android应用程序时采用Java语言编写,程序在Android运行时中执行,其运行时分为核心库和Dalvik虚拟机两部分。

(1) 核心库。核心库提供了Java语言API中的大多数功能,同时也包含Android的一些核心API,如android.os、android.net、android.media等。

(2) Dalvik虚拟机。Android程序不同于J2ME程序,每个Android应用都运行在自己的进程上,享有Dalvik虚拟机为它分配的专有实例,并在该实例中执行。Dalvik虚拟机是一种基于寄存器的Java虚拟机,而不是传统的基于栈的虚拟机,并进行了内存资源使用的优化以及支持多个虚拟机的特点。Java编译器将Java源文件转为class文件,class文件又被内置的dx工具转化为dex格式文件,该格式文件针对最小内存使用做了优化,这种文件在Dalvik虚拟机上注册并运行。在一些底层功能方面,例如线程和低内存管理等,Dalvik虚拟机是依赖Linux内核的。

4. Linux 内核

Android是基于Linux 2.6内核,由C语言实现。其核心系统服务如安全性、内存管理、进程管理、网路协议以及驱动模型都依赖于Linux内核。内核部分还相当于一个介于硬件层和系统中其他软件组之间的一个抽象层次。

除了标准的Linux内核外,Android系统还增加了Binder IPC驱动、WiFi驱动、蓝牙驱动等驱动程序,为系统运行提供了基础性支持。

1.3 Android 环境搭建

Android软件开发包可以在Windows(如Windows XP、Vista或Windows 7)、Linux和Mac OSX上安装运行。然后将在这些操作系统中开发的Android应用程序部署到任意的Android设备上。

Android软件开发包主要包括JDK、Eclipse和Android SDK,在进行应用程序开发之前,首先必须将开发环境搭建起来。本书介绍的是在Windows操作系统下搭建开发环境,执行开发编程。

1.3.1 下载 Android 开发工具

要在Windows操作系统上开发Android应用项目,必须要有三个工具软件:①JDK,Android主要使用Java语言来开发应用程序,所以必须要有JDK开发包;②Eclipse,Eclipse是常用的Java语言的集成开发环境;③SDK,SDK是Software Development Kit的缩写,是

专门用于开发 Android 应用的软件开发工具包。

JDK、Eclipse 和 SDK 这三个工具软件可以从一些专业网站中免费下载。由于现代软件行业的发展变化，以及这些软件版本的不断升级，其下载网址和文件名，以及安装配置过程都有可能会发生变化。本书所介绍的工具软件及开发环境是至 2012 年 6 月，最新的下载软件及环境搭建，供大家在实践中参考。

1. 下载 JDK

从网址为 http://www.oracle.com/technetwork/java/javase/downloads/index.html 的网页中，下载 Java SE 的开发工具包 JDK，如图 1-3 所示。

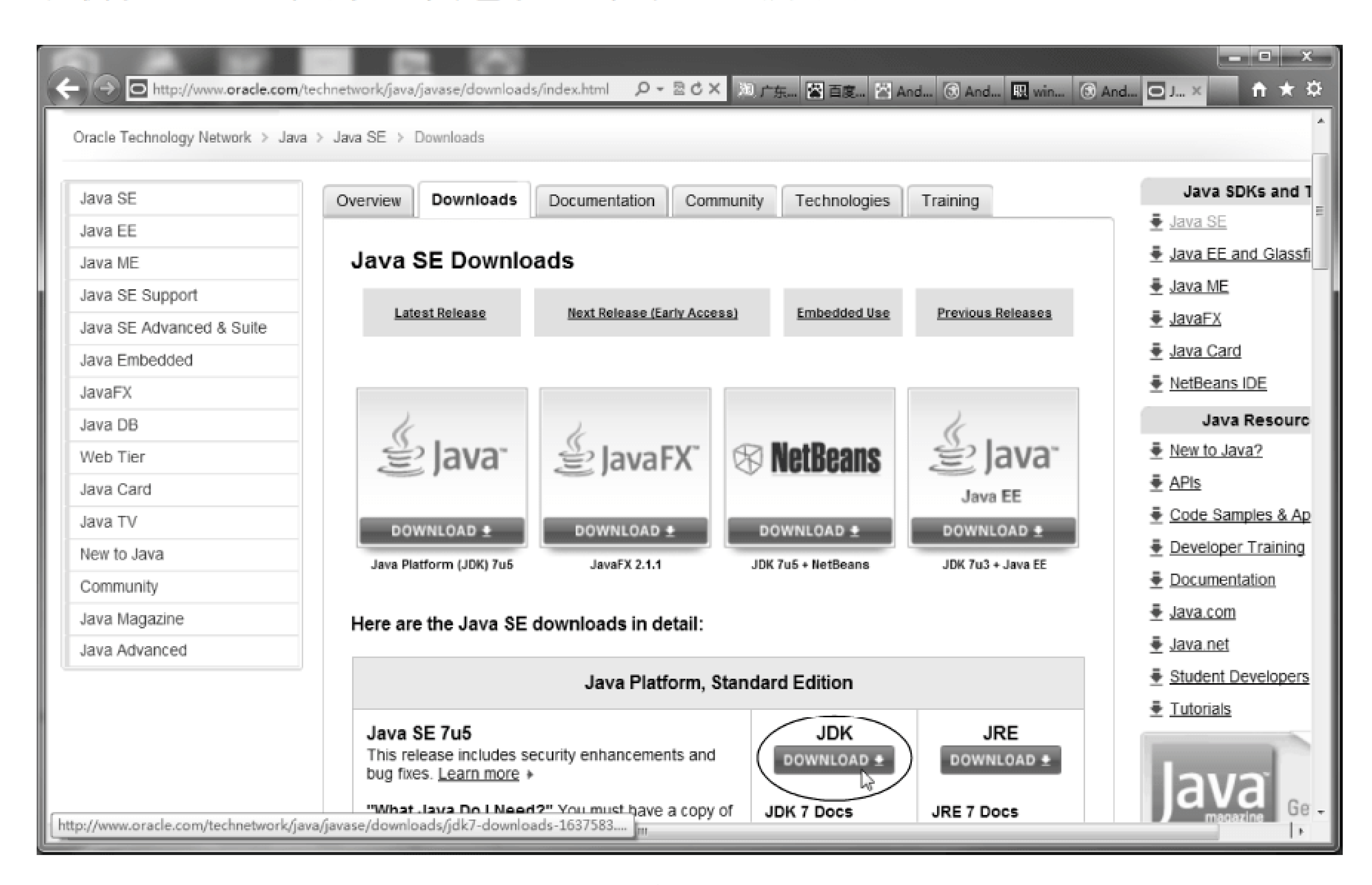


图 1-3 Oracle 技术网站的 Java 下载网页

选择在 Java Platform, Standard Edition 中的 Java SE 7u5 的 JDK 下载，即单击如图 1-3 所标记的 DOWNLOAD 按钮，进入下载细目页。在下载细目中选择适用 Windows 的文件下载：如果计算机操作系统是 32 位的，下载 jdk-7u5-windows-i586.exe；如果计算机操作系统是 64 位的，下载 jdk-7u5-windows-x64.exe。本书选择的是下载 jdk-7u5-windows-i586.exe 文件，保存在本机的文件夹中。

2. 下载 Eclipse

在网址为 http://www.eclipse.org/downloads/的网页中，选择要下载的 Eclipse 工具。如果是开发不用配置 Tomcat 网络服务器的应用程序，可以选择 Eclipse IDE for Java Developers 项下载，如果应用程序需要使用到 Tomcat 网络服务器，建议选择 Eclipse IDE for Java EE Developers 项。本书选择后者，单击 Eclipse IDE for Java EE Developers 项进入下载细目网页，如图 1-4 所示。

Eclipse 有不同的下载包，如图 1-4 左边窗口中列出的 Indigo、Helios、Galileo 等。本书选择的是 Helios Packages。单击 Helios Packages，进入文件下载页，选择 Eclipse IDE for Java EE Developers 项后面的 Windows 32-bit，进入如图 1-5 所示的下载镜像页，然后单击下载图标，即可下载 eclipse-jee-helios-SR2-win32.zip，将该文件保存在本机的文件夹中。

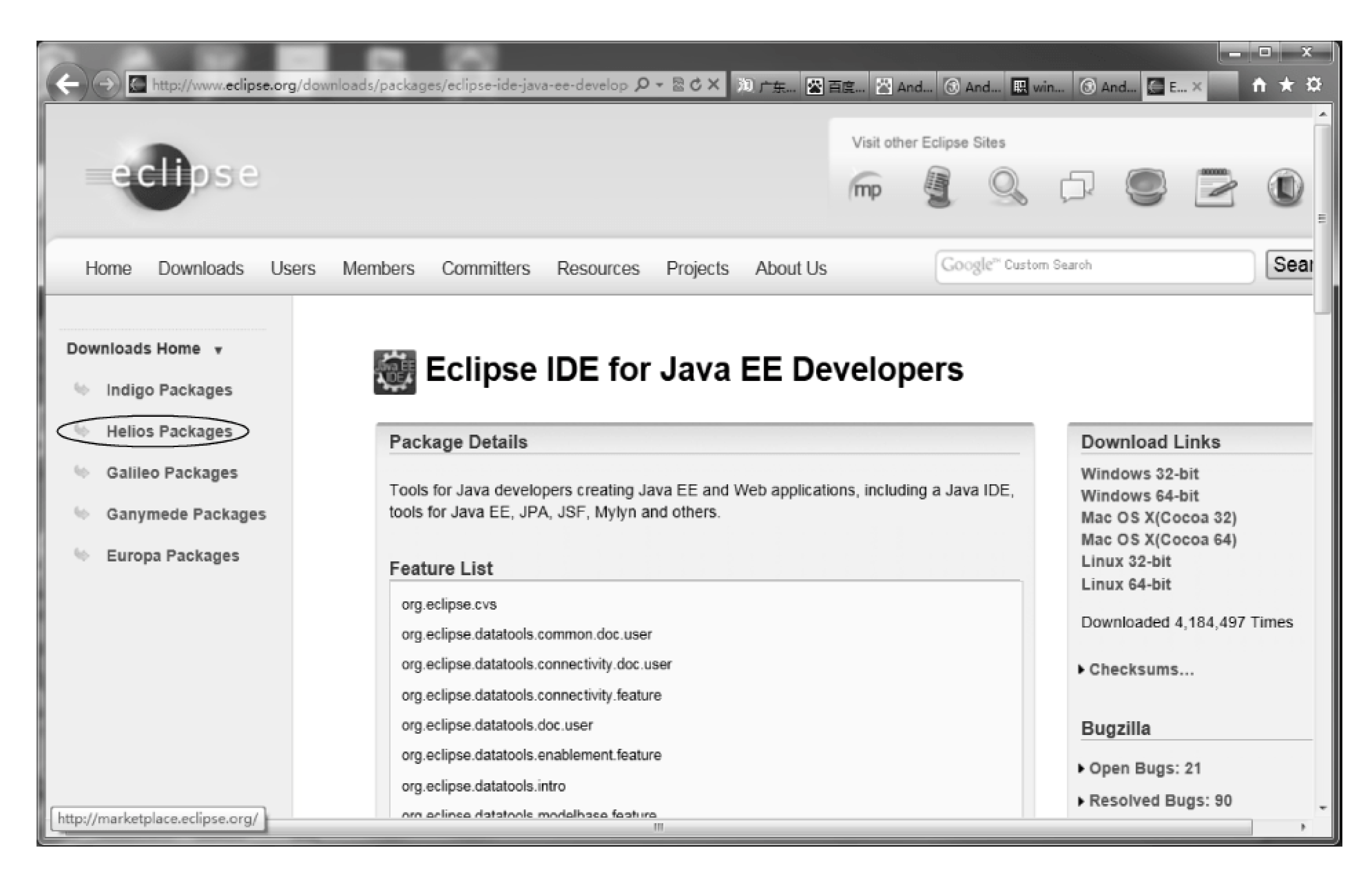


图 1-4　Eclipse IDE for Java EE Developers 开发工具的下载网页



图 1-5　Eclipse 的下载镜像页

3. 下载 SDK

输入网址 http://developer.android.com/sdk/index.html 进入 Android 的开发工具网页,如图 1-6 所示。单击右边窗口的 Download the SDK for Windows 项,即可下载或运行来自 dl.google.com 的 installer_r18-windows.exe。本书选择的是保存该文件到文件夹中。

现在已经将需要的三个工具软件下载完毕,保存在 E:\3gms\android_downloads\2012 文件夹中,如图 1-7 所示。

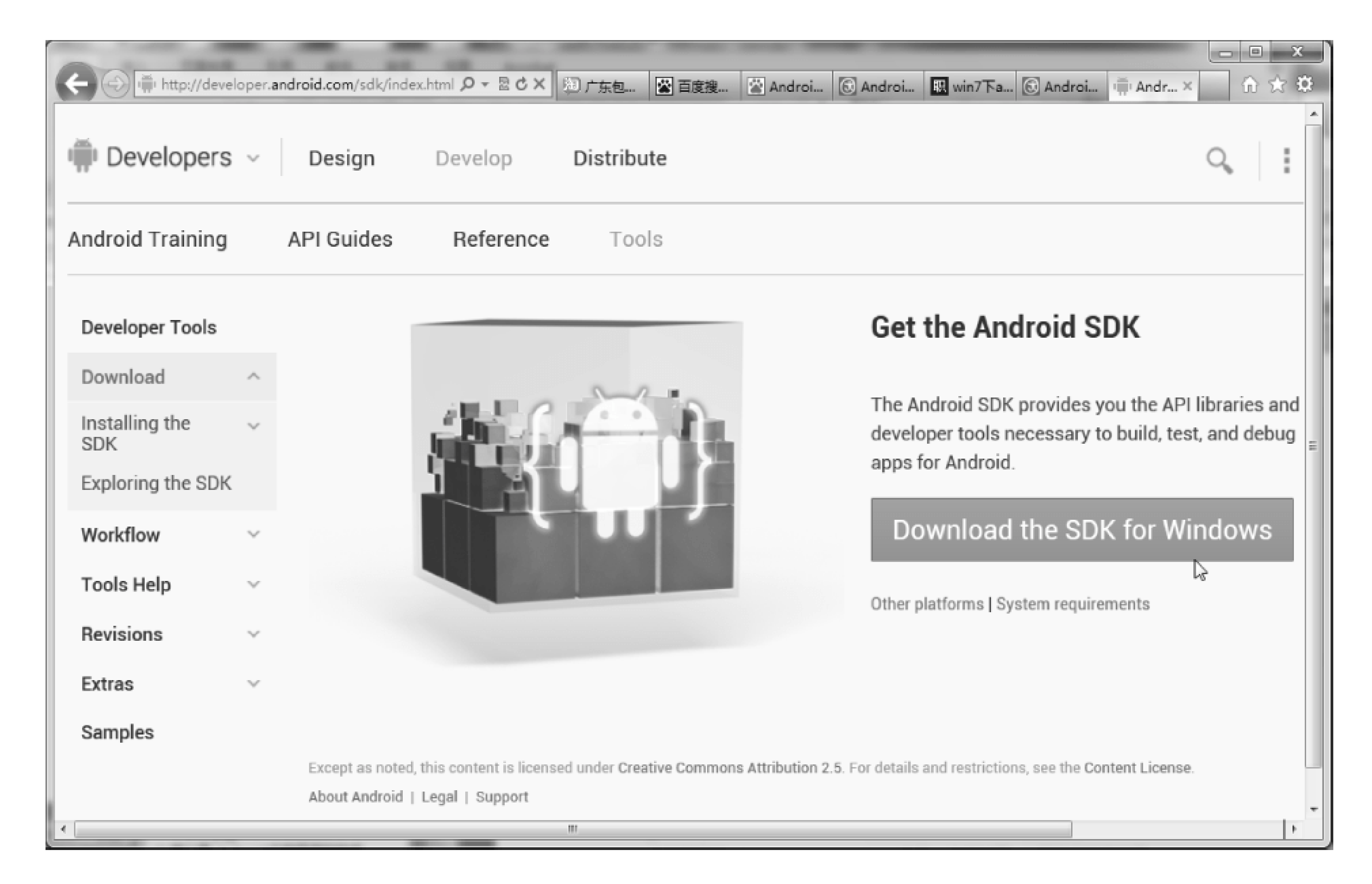


图 1-6 Android 开发工具网页

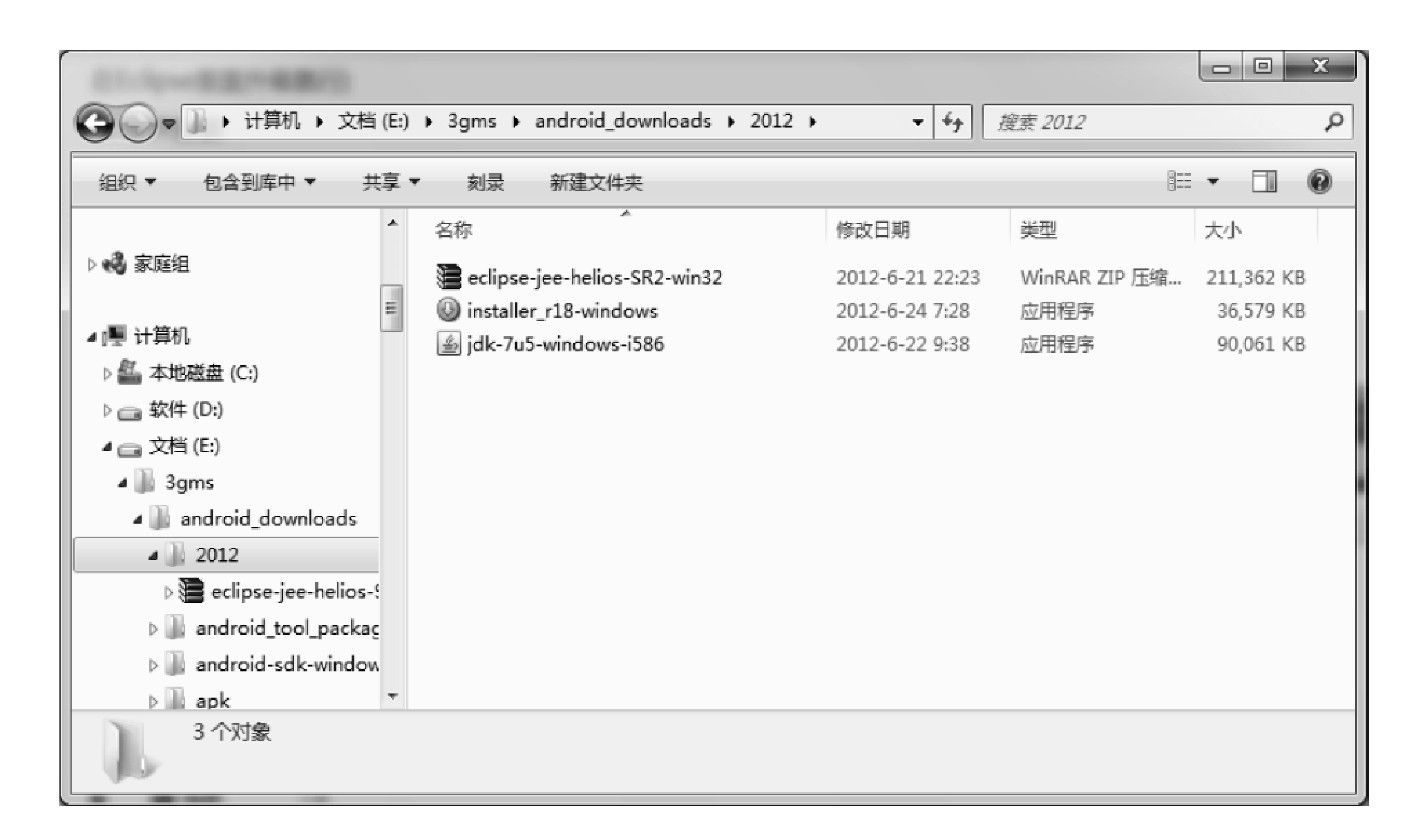


图 1-7 下载的软件存放的位置

1.3.2 开发环境的安装与配置

1. 安装 JDK

运行下载的 jdk-7u5-windows-i586.exe 文件，出现如图 1-8 所示的安装向导。

按照向导逐步完成安装。本书中将其安装在 C:\Program Files\Java\jdk1.7.0_05 文件夹中。为了检查 JDK 安装是否成功，可使用如下方法进行检验。



图 1-8 安装 JDK 的向导

(1) 单击"开始"菜单，选择"运行..."菜单项，在"运行"对话框中输入"cmd"命令，进入命令行状态。

(2) 在命令行状态提示符后输入命令："java -version"，然后按 Enter 键。如果得到如图 1-9 所示的信息，则表示安装成功。

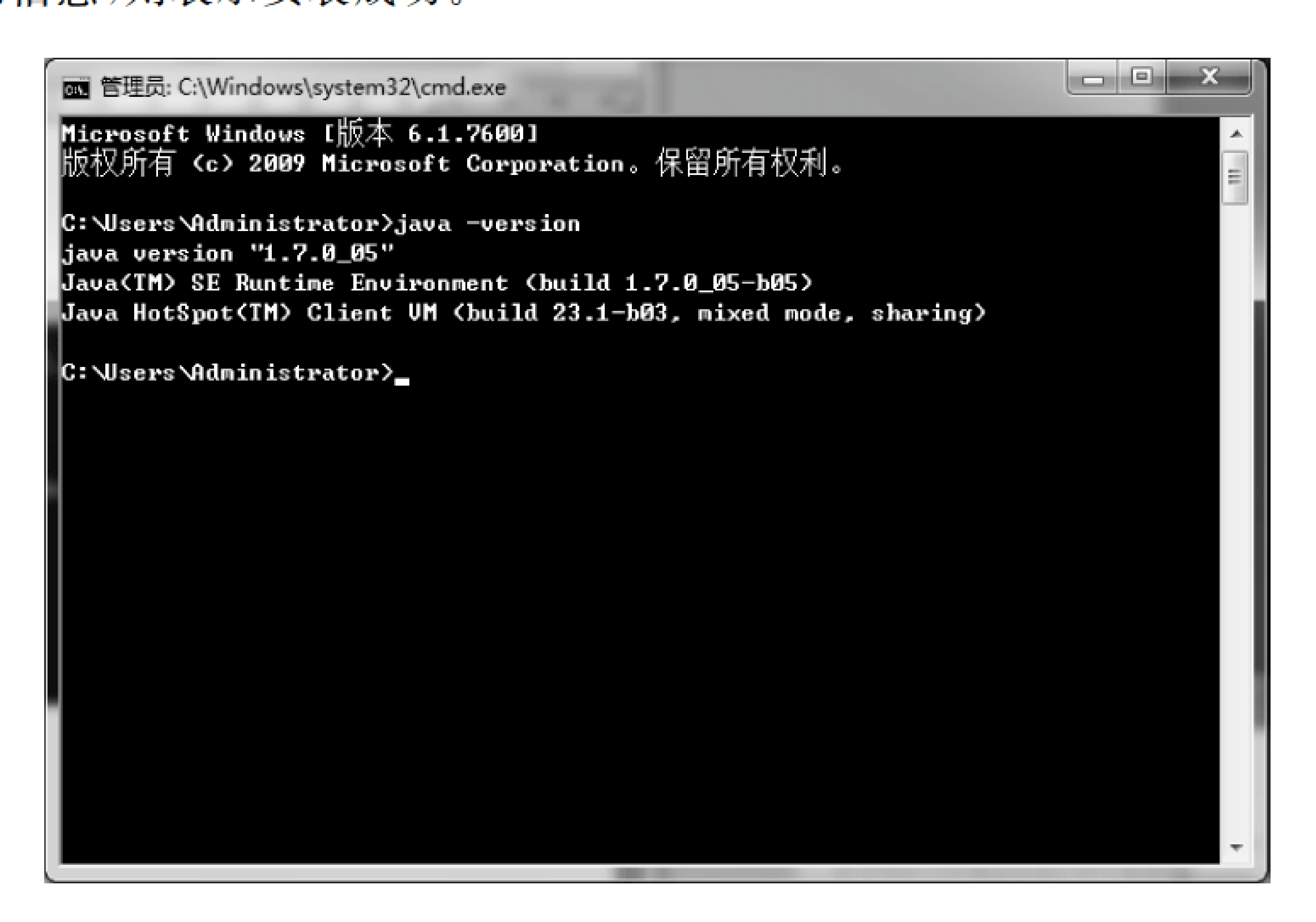


图 1-9 检验 JDK 安装的成功信息

2. 安装 Eclipse

Eclipse 不需要安装，直接解压 eclipse-jee-helios-SR2-win32.zip 文件到指定的文件夹中即可。本书将其解压在 E:\Eclipse 文件夹下。

3. 安装 SDK

运行下载的 installer_r18-windows.exe 文件，可能会出现一个无法验证发布者的安全警告窗，这个程序来自于 Google，单击"运行"按钮，进入安装向导，如图 1-10 所示。

单击 Next 按钮进入安装页，在安装 SDK 前，程序会先检测计算机是否已安装了 JDK。

在前面已经安装好了 JDK，所以单击 Next 按钮，逐步按照向导提示进行操作，直到完成，如图 1-11 所示。本书的 SDK 安装路径为 E:\android-sdk。

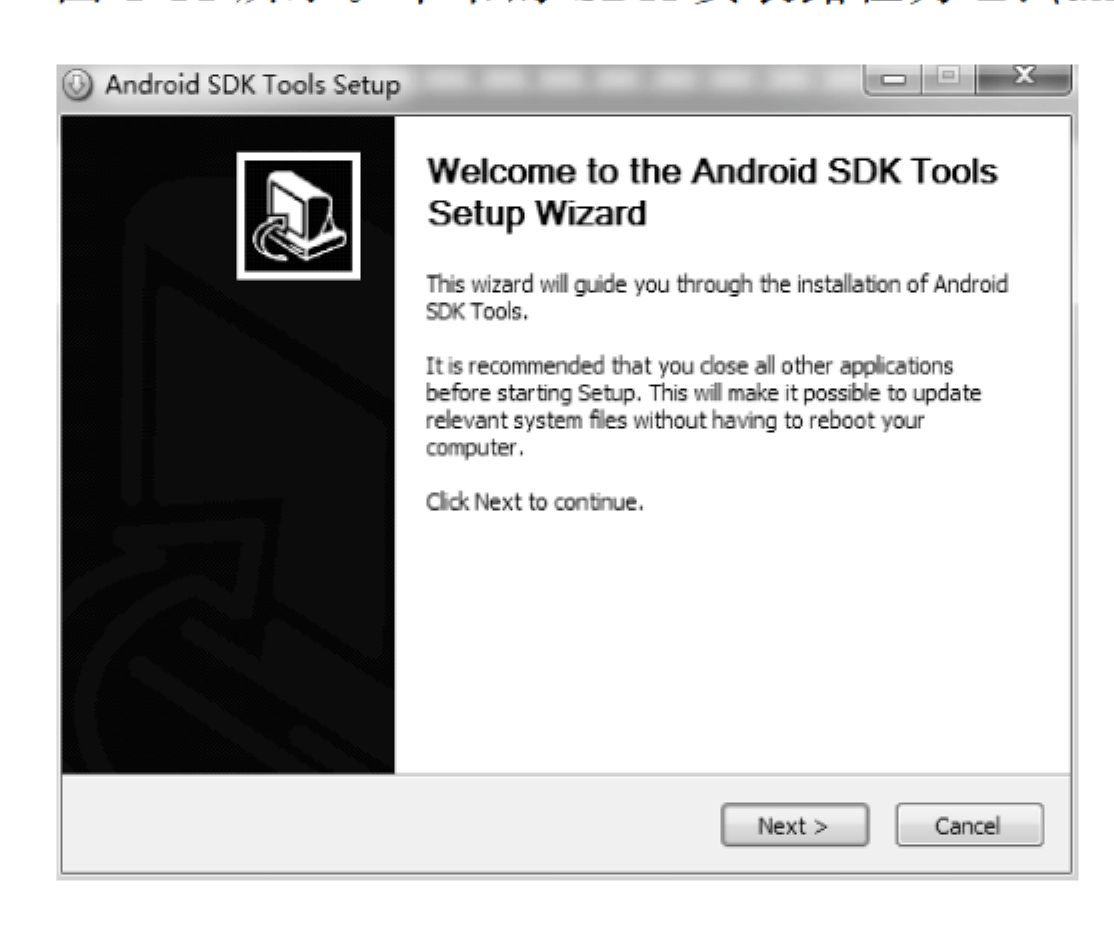


图 1-10　安装 SDK 的向导

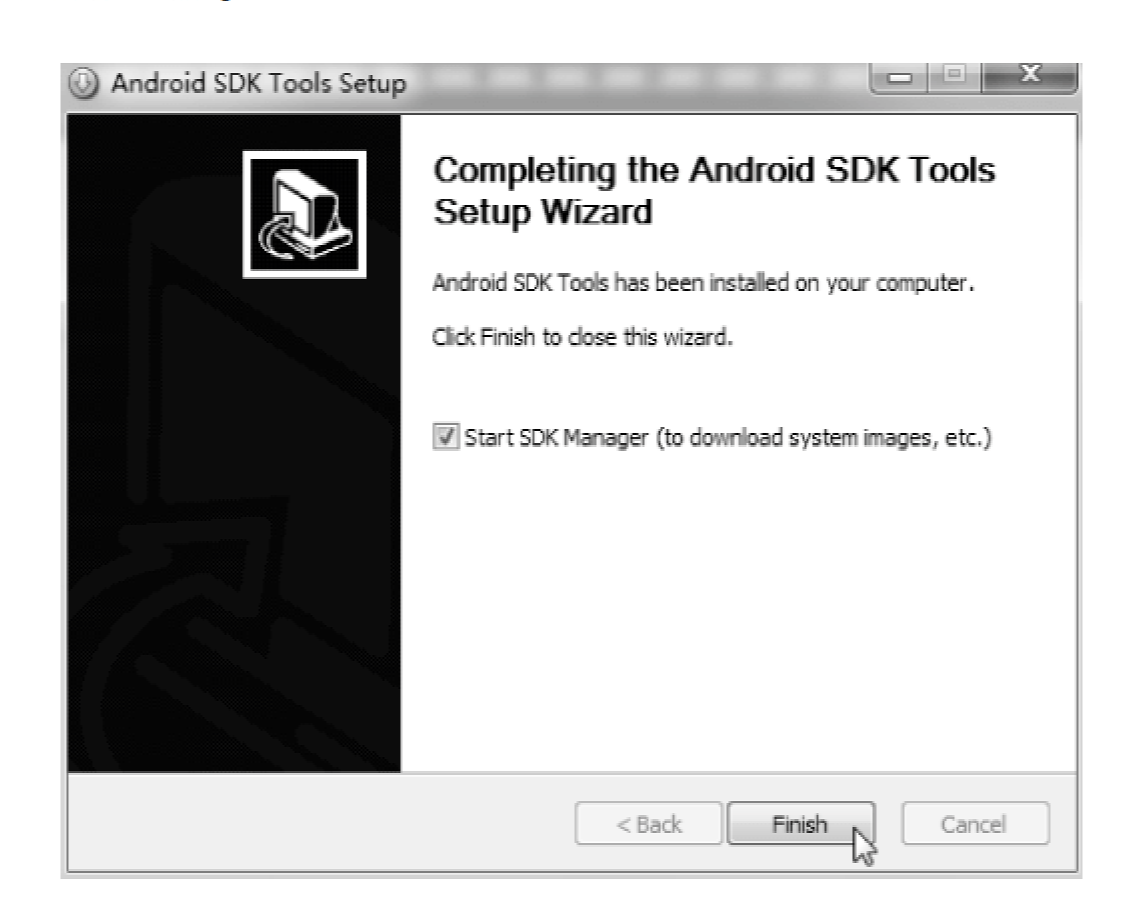


图 1-11　安装完成

在完成安装 SDK 之后，需要对该 Android SDK 兼容的各种版本进行下载、升级和更新，所以在图 1-11 中勾选了 Start SDK Manager(to download system images, etc.)，然后单击 Finish 按钮。然后进入 SDK 管理程序，如图 1-12 所示。

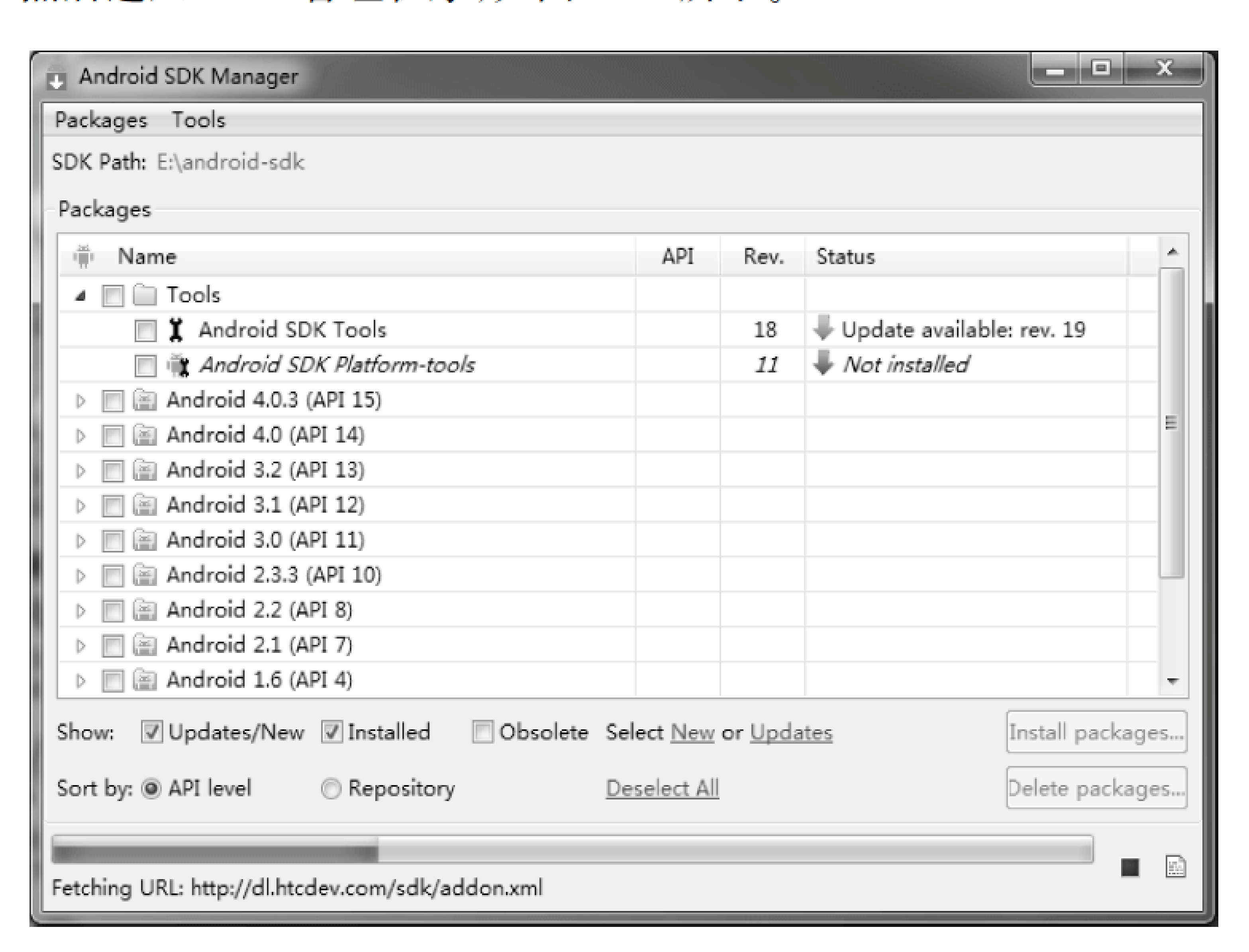


图 1-12　Android SDK Manager 对话框

SDK Manager 首先会搜索一遍本机中的 SDK 版本的安装情况，待搜索完毕，可以看到原来的 Install packages...按钮变成了 Install 60 packages...，这说明有 60 个安装包需要从网络上下载安装，当然，可以通过勾选或不勾选部分 Android SDK 版本的项目，此时按钮中的数字会随之变化。单击该按钮进入 Choose Packages to Install 对话框，如图 1-13 所示。

在此选择 Accept All 单选按钮，当然，也可以选择 Accept 或其他单选按钮，然后单击 Install，进行下载和安装。在安装过程中有时会弹出一些对话框，可以选择继续下去，有些可能需要输入密码等信息，这是因为 SDK 3.0 版本以后，有些资源不是完全开放的，这时一般

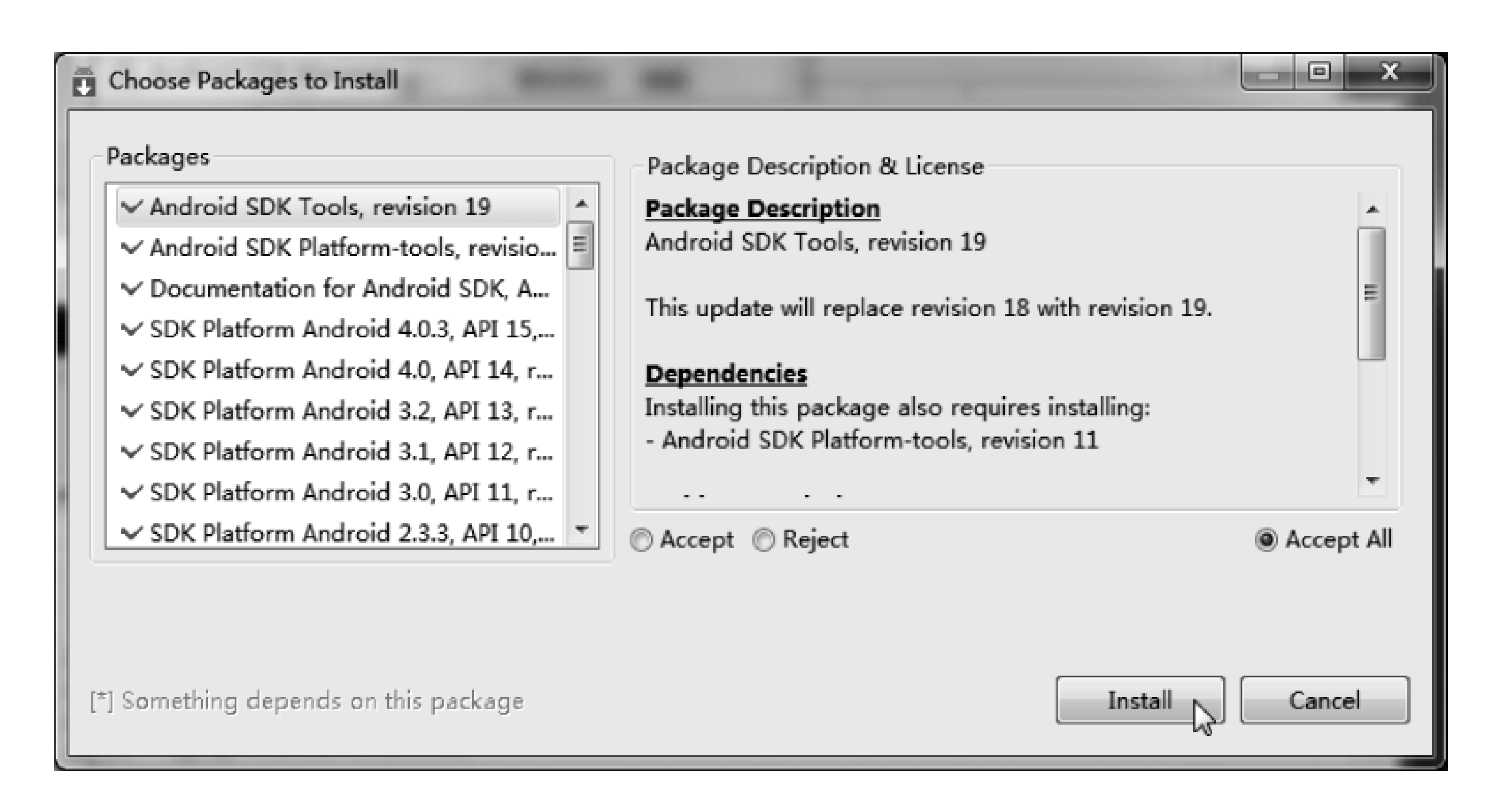


图 1-13 Choose Packages to Install 对话框

可以不安装这些 packages。注意，这个过程持续时间比较长，请耐心等待，并保持网络的畅通。

安装完毕后，请关闭 Android SDK Manager 窗口，然后再次打开它，看看需要安装的包是否已经成功安装。

4. 安装 Eclipse 插件 ADT

这时的 Eclipse 还不能开发 Android 应用程序，必须在安装了 ADT（Android Development Toolkit）插件后才能进行 Android 应用开发。ADT 插件在 Eclipse 中集成的功能有：新建项目向导，并且包含基本的应用向导，基于表单的 manifest、layout 和 resource 编辑器，自动编译 Android 项目，Android 模拟器，Dalvik 调试监控服务（DDMS），访问设备或模拟器的文件系统，运行时调试，所有的 Android/Dalvik 日志和控制台输出等。所以，想要在 Eclipse 中开发 Android 应用程序，需要安装 ADT 插件。

其操作步骤如下。

（1）启动 Eclipse。

注意，首次启动 Eclipse 时会出现一个加载工作空间的对话框，如图 1-14 所示。

图 1-14 设置项目的工作路径

在这个对话框中，可以设置应用项目的工作路径，如果不想每次启动 Eclipse 都出现此对话框，可以勾选 Use this as the default and do not ask again 项。然后单击 OK 按钮，首次启动 Eclipse 时会出现欢迎界面，如图 1-15 所示。以后再次启动 Eclipse 就不会出现这个欢迎界面，而是直接进入 Eclipse 工作界面，如图 1-16 所示。

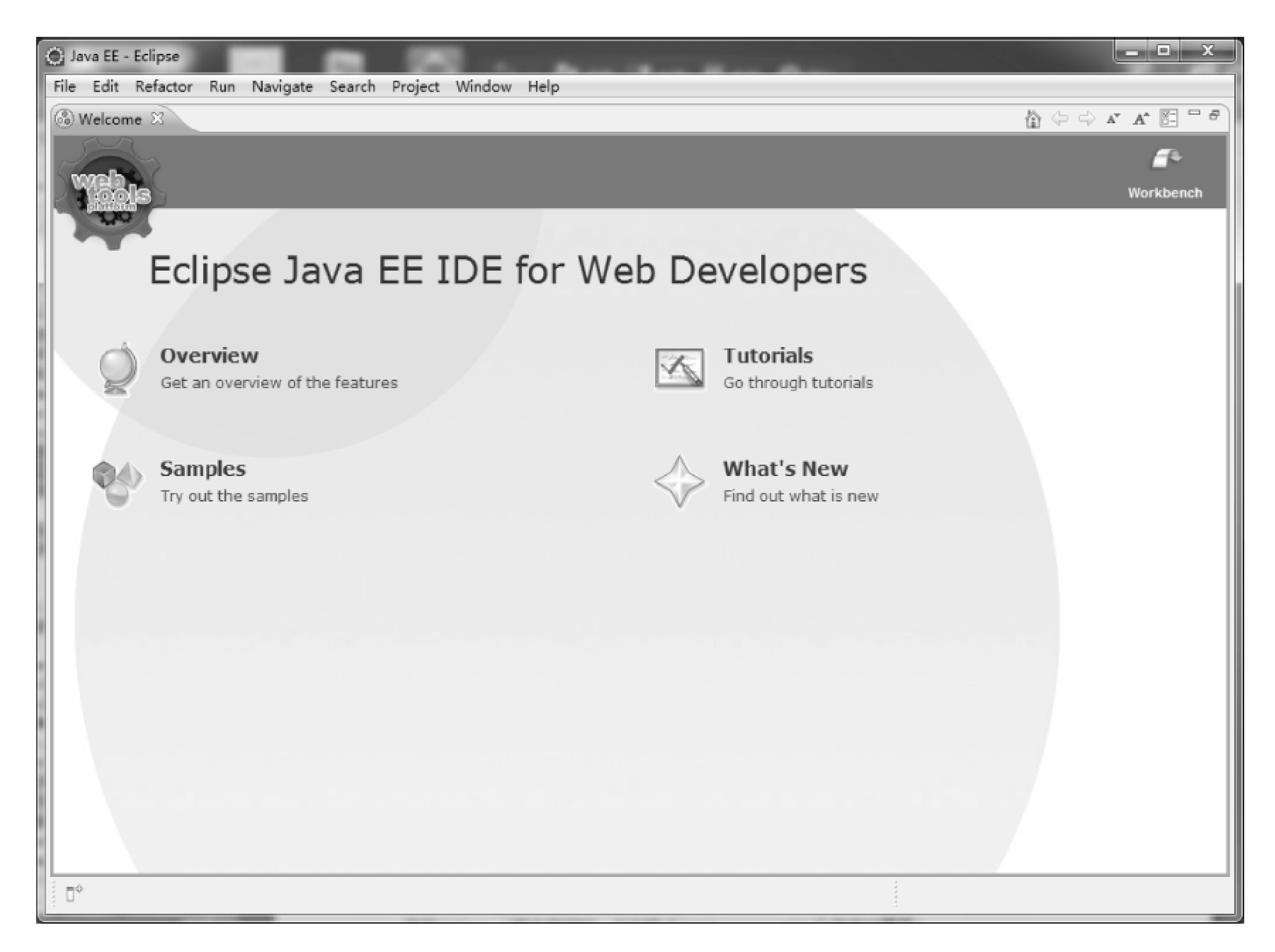


图 1-15 Eclipse 的欢迎界面

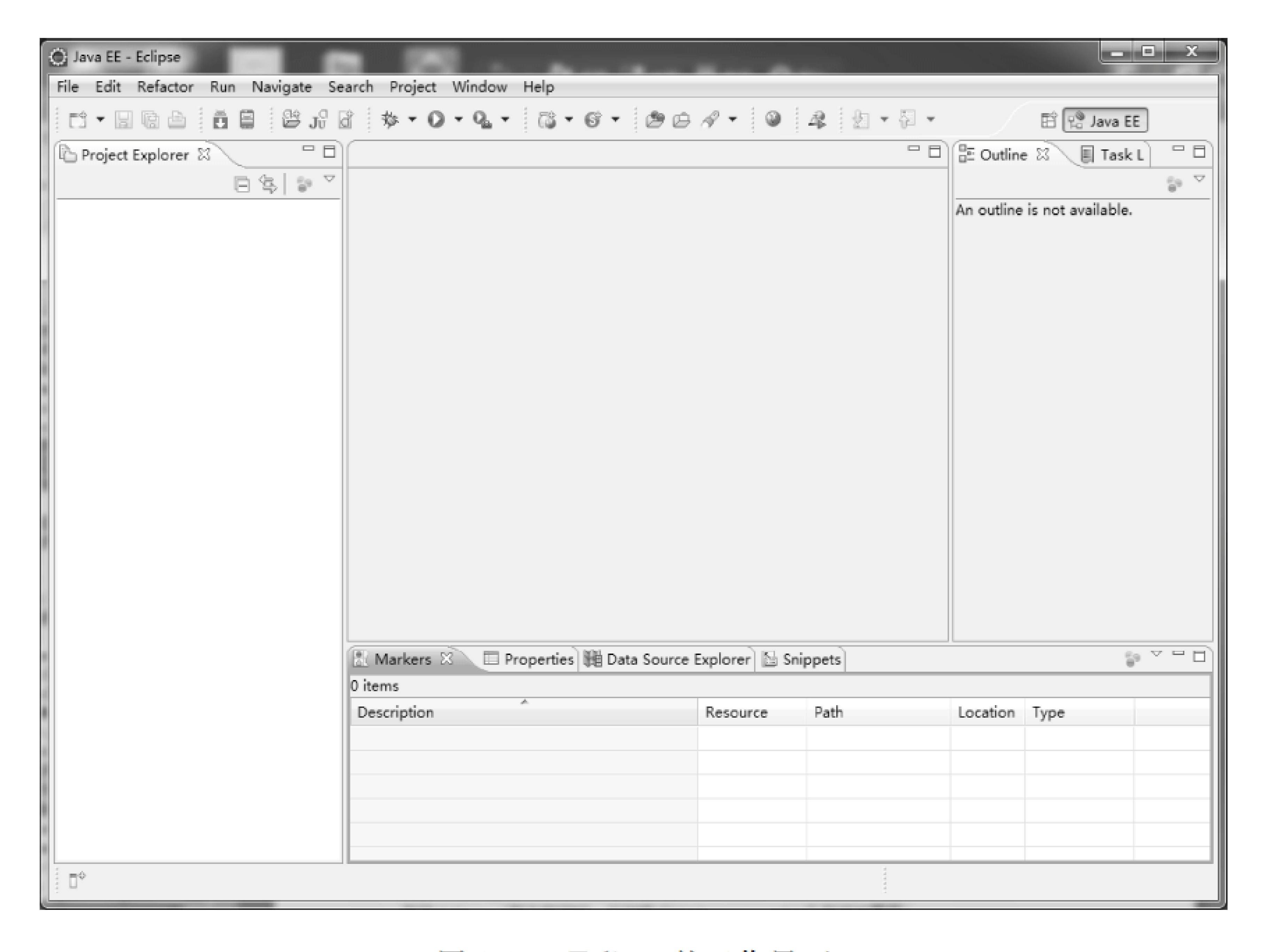


图 1-16 Eclipse 的工作界面

（2）添加 Site。

选择菜单 Help → Install New SoftWare...，进入 Install 对话框，如图 1-17 所示。

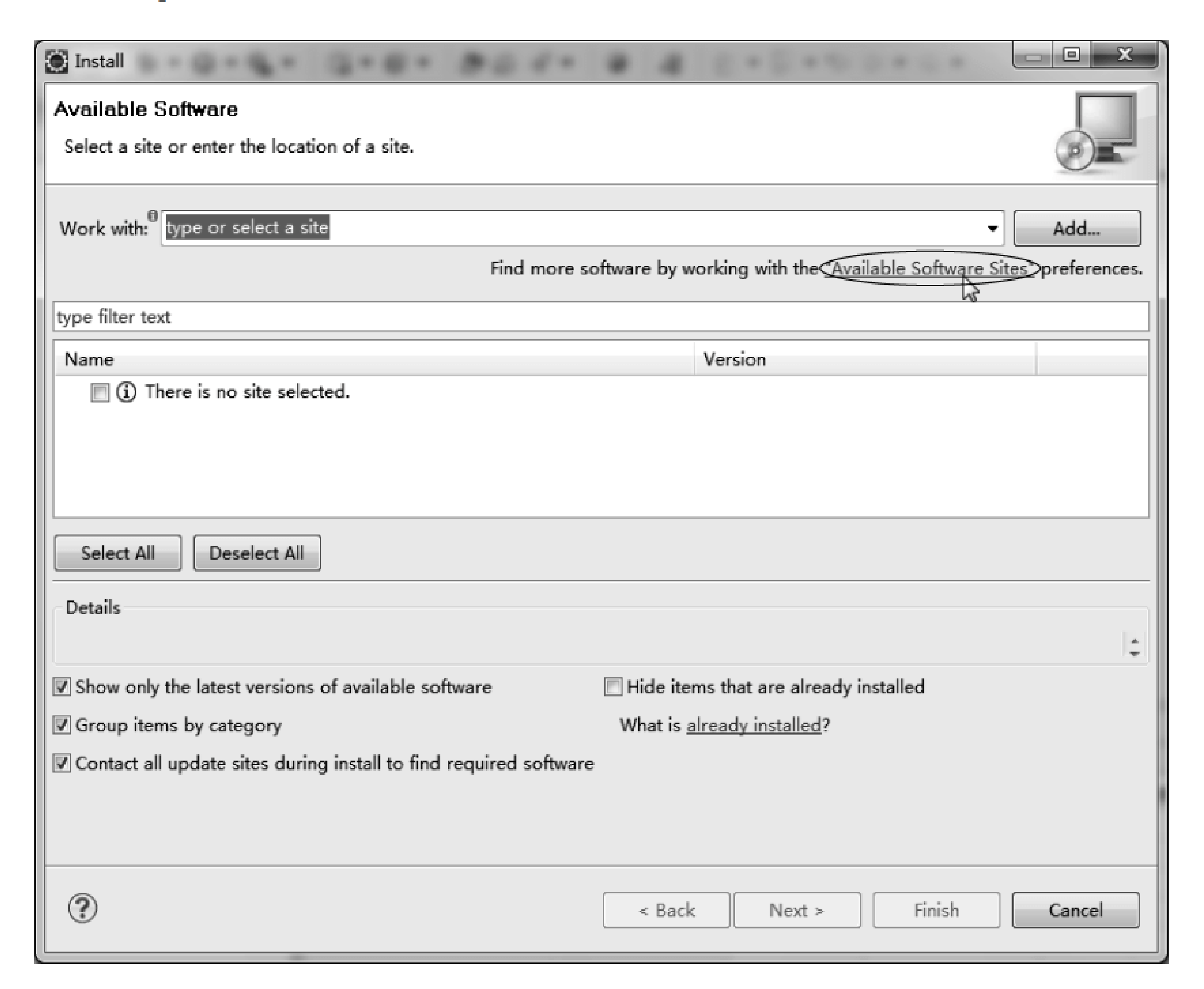


图 1-17　Install 对话框

单击图 1-17 中标记处的 Available Software Sites，进入 Preferences 对话框，如图 1-18 所示。

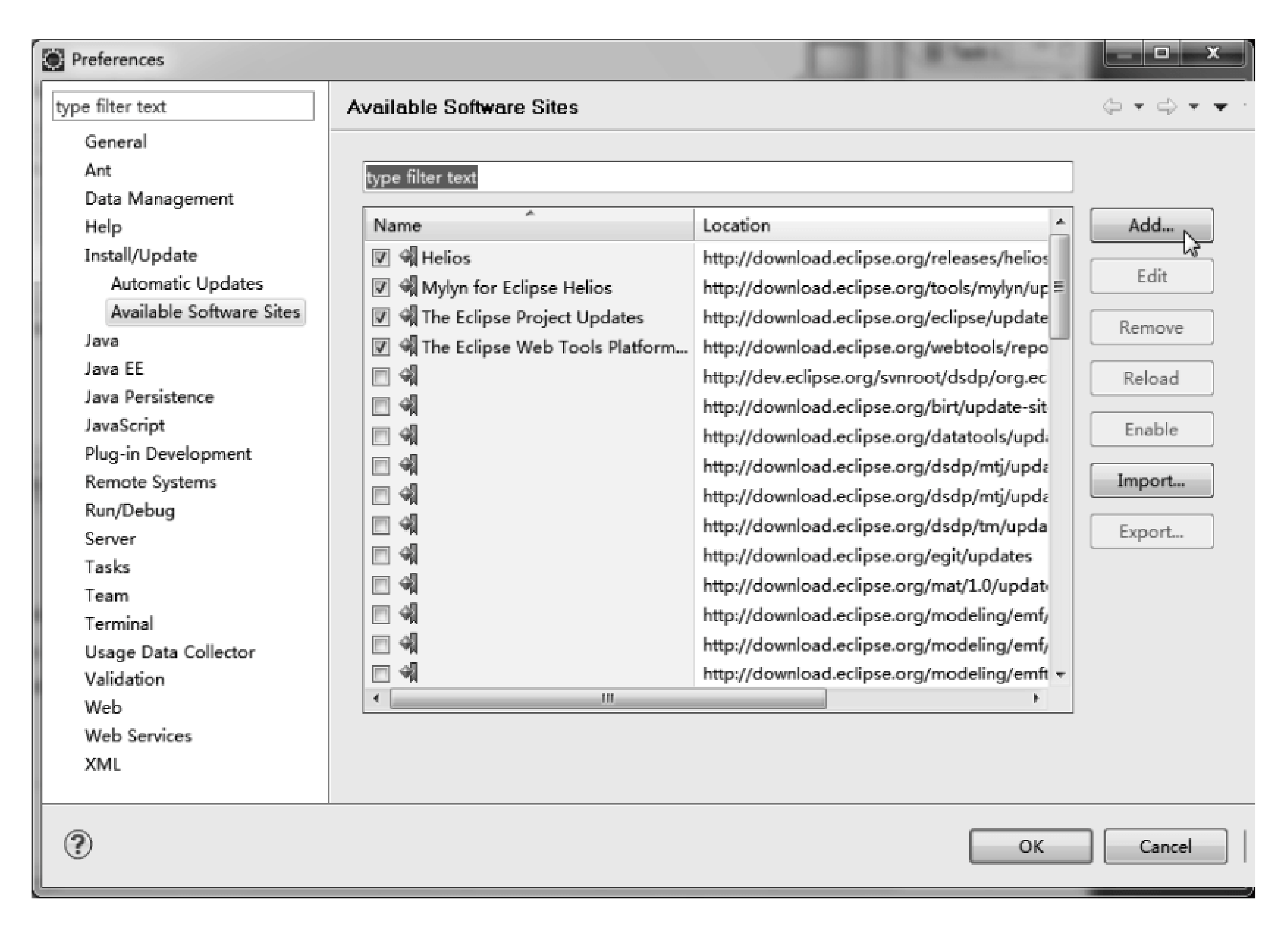


图 1-18　Preferences 对话框

单击 Add 按钮，进行添加可用的下载插件的站点 Site 设置，如图 1-19 所示。

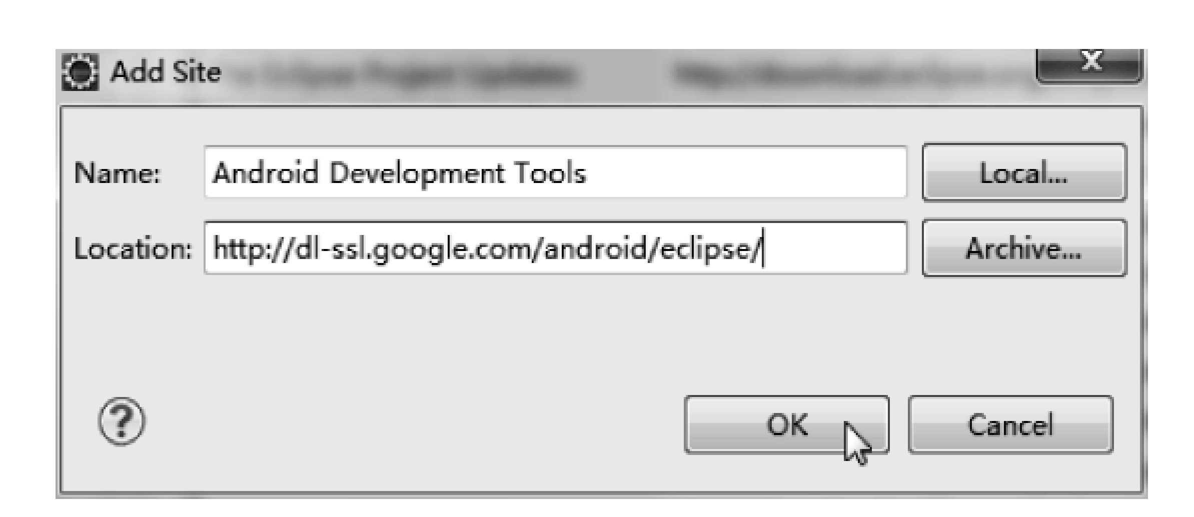


图 1-19 Add Site 对话框

在此对话框中输入站点的名字，该名字由用户自己定义，可以随便命名。在这里输入“Android Development Tools”，该站点所在的网址“http://dl-ssl.google.com/android/eclipse”，然后单击 OK 按钮，即可以在 Preferences 对话框的右边看到新添加的 Site 了。单击 Preferences 对话框中的 OK 按钮返回到 Install 对话框。

（3）下载并安装插件。

在 Install 对话框的 Work with 下拉列表中选择刚创建的 Site 项，然后在 type filter text 的下面，展开 Developer Tools 可以看 Android DDMS、Android Development Tools、Android Hierarchy Viewer 等项目，勾选它们，如图 1-20 所示。

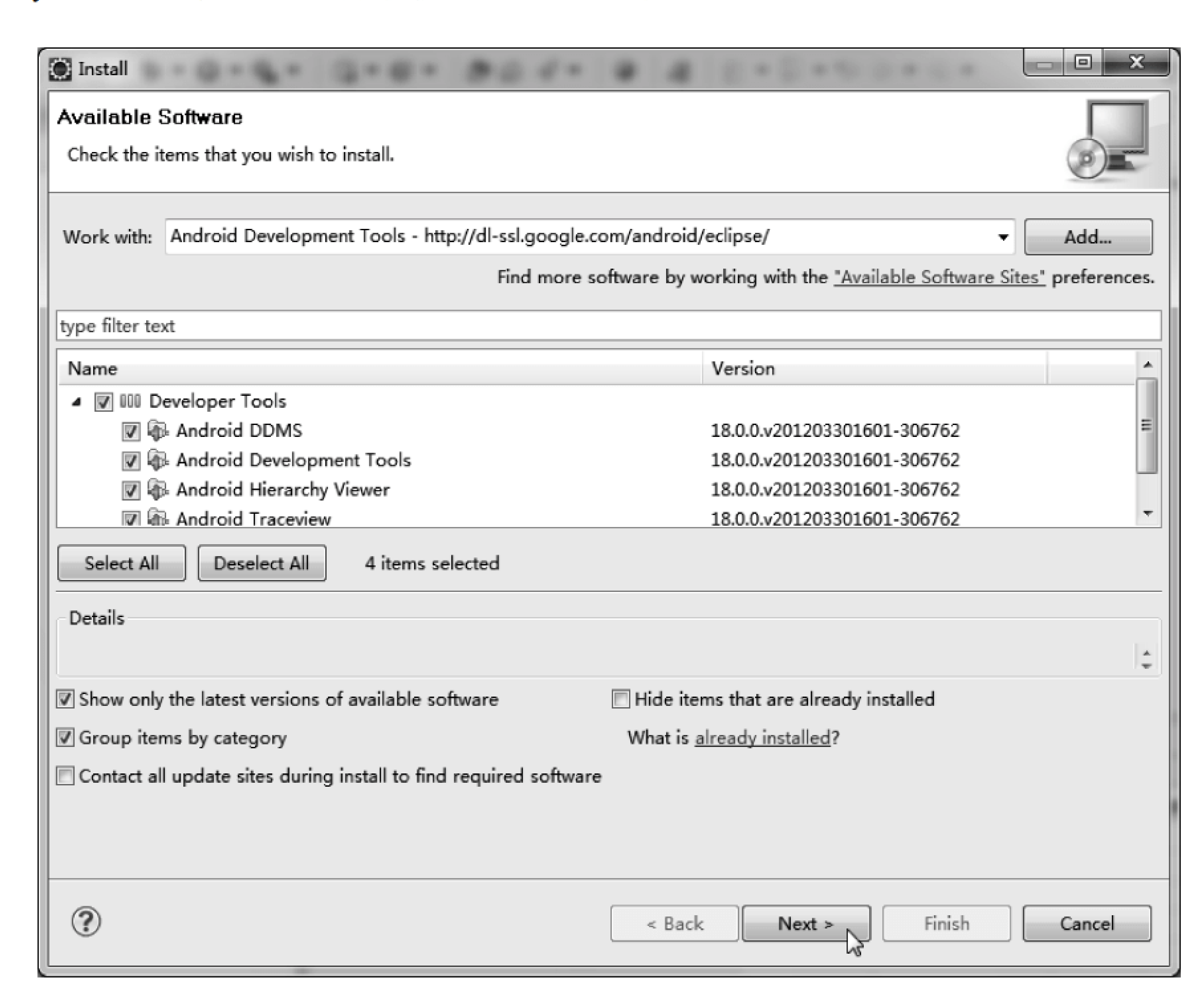


图 1-20 选择 Work with 项，勾选 type filter text 下面的所有项

单击 Next 按钮，进入显示所要安装插件的细节内容的页面，如图 1-21 所示。

继续单击 Next 按钮，进入确认安装插件页面，如图 1-22 所示。

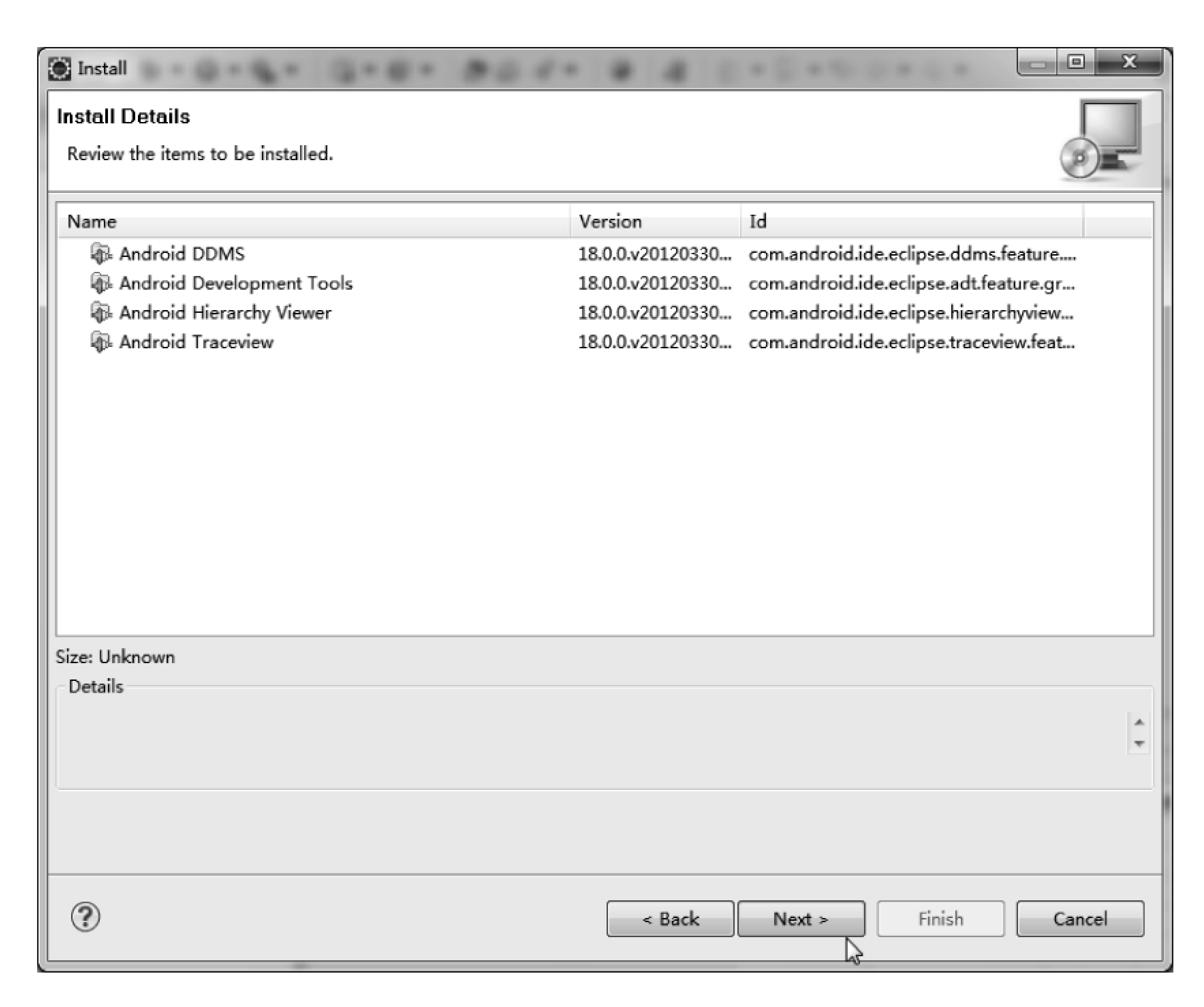


图 1-21　显示所要安装的插件细节页面

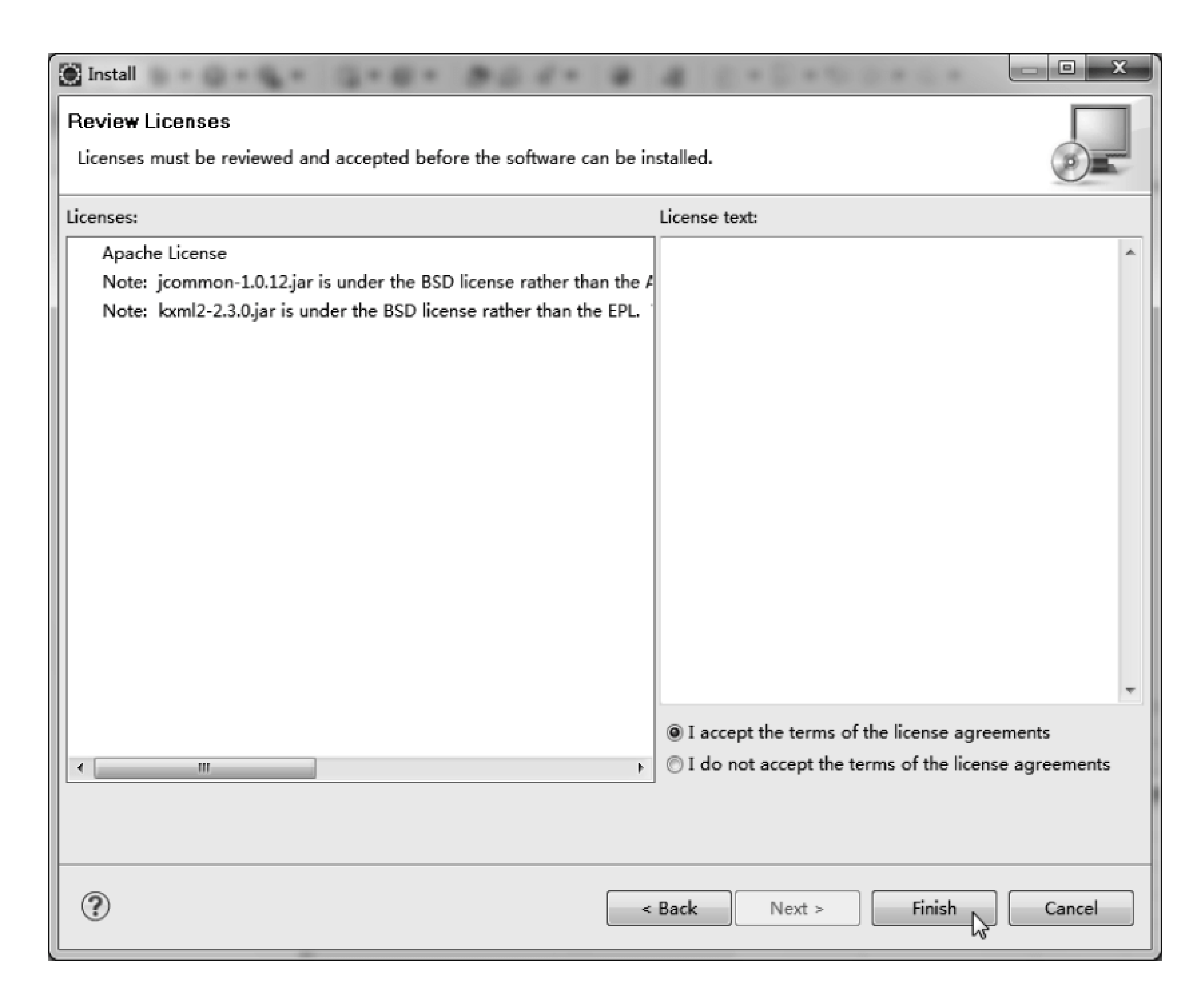


图 1-22　确认安装插件页面

选择 I accept the terms of the license agreements 单选项，然后单击 Finish 按钮，开始下载和安装 ADT 插件。在安装过程中会弹出一些警告提示框，单击 OK 按钮继续安装。

安装完成后，出现重新启动 Eclipse 对话框，如图 1-23 所示。

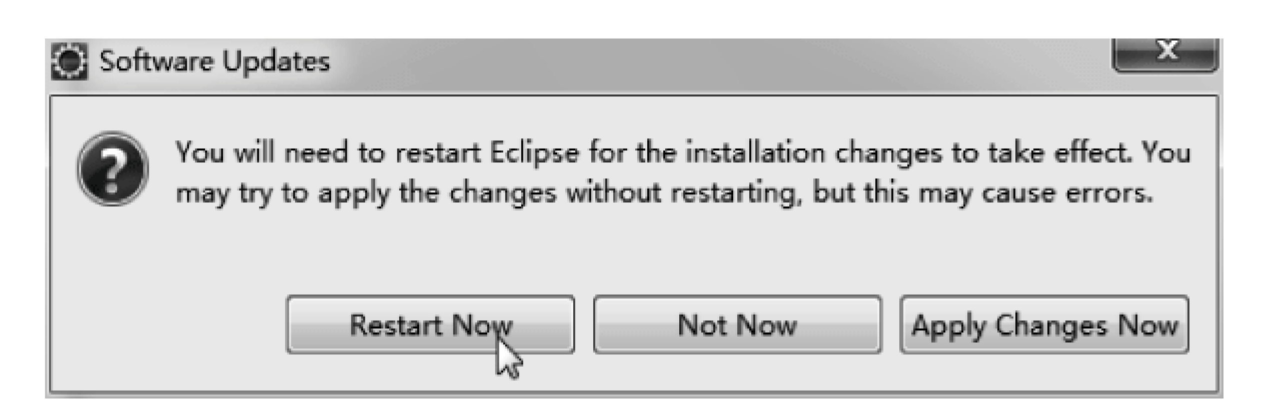


图 1-23 安装完成后的对话框

单击 Restart Now 按钮，立即执行重启 Eclipse 操作。

5. 为 Eclipse 指认 SDK

这一设置的目的是为了告诉 Eclipse，Android 的 SDK 位于何处。在安装 ADT 插件后，会重启 Eclipse。重启 Eclipse 后系统会自动地完成指认 SDK 的路径，如图 1-24 所示。如果此时 SDK 中有需要升级的版本包，系统还会自动运行 SDK Manager 程序进行升级安装。

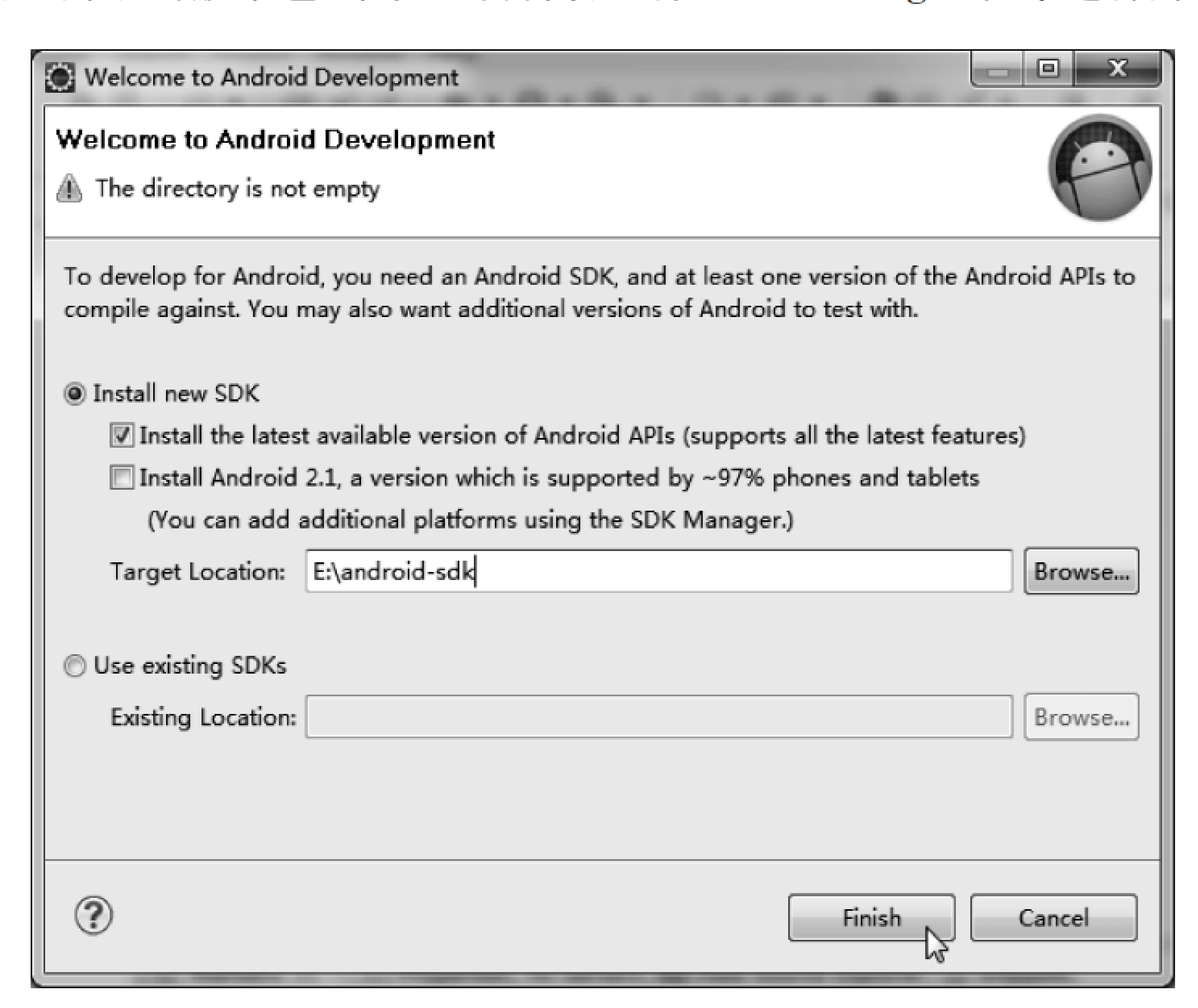


图 1-24 自动指认 SDK 的位置

如果重启 Eclipse 时没有自动做上述操作，可以手动进行设置，其操作步骤如下。

(1) 启动 Eclipse。如果已经启动了 Eclipse，可以忽略这一步。

(2) 指出 SDK 所在位置。选择菜单 Window→Preferences，在左侧窗口的列表中选择 Android，这时右侧窗口显示有关 Android 的 Preferences。在 SDK Location 输入框内输入 SDK 所在的文件夹路径，当然也可以通过单击 Browse...按钮来选择 SDK 所在的文件夹路径。本书的 SDK 的保存路径是 E:\android-sdk，如图 1-25 所示。

6. 为 Eclipse 创建 AVD

在 Eclipse 中运行 Android 应用程序，必须要在 Eclipse 中创建 AVD(Android Virtual

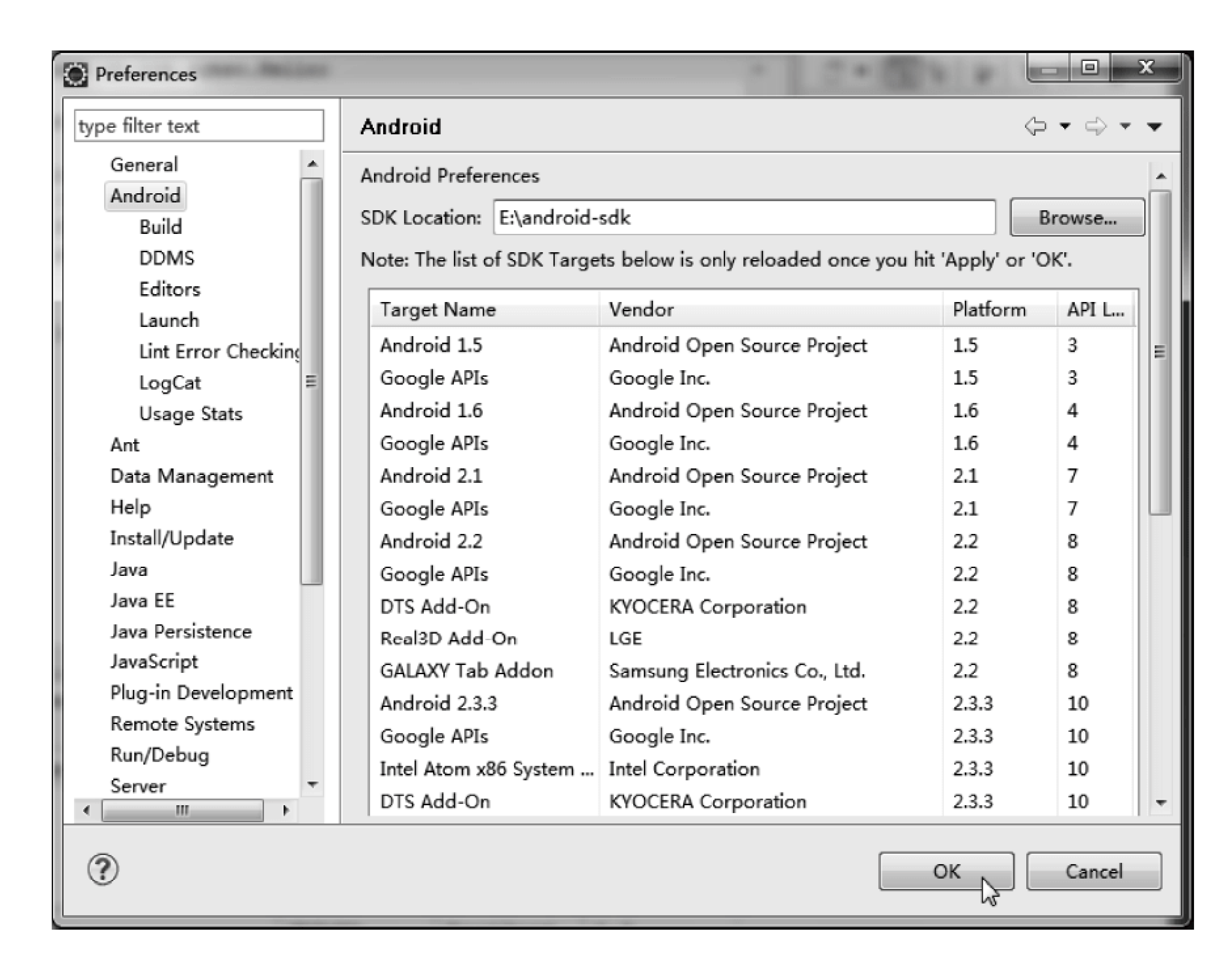


图 1-25 为 Eclipse 指出 SDK 的所在位置

Device，Android 虚拟设备)，每个 AVD 模拟一套虚拟设备来运行 Android 应用程序。AVD 又称为 Android 模拟器，占用 PC 上的硬盘空间，可以模拟不同款手机设备，运行效果与手机真机几乎相同。可以创建一个 AVD，也可以创建多个 AVD。

创建 AVD 的方法有两种。一是通过命令行创建，二是通过 Eclipse 开发环境创建。

(1) 使用命令创建 AVD。在"开始"菜单的"运行..."菜单项中输入命令"cmd"，进入命令行状态，在提示符后输入以下命令创建 AVD，命令的格式为：

```
android create avd -- name <名字> -- target <版本 ID 号>
```

例如，要创建一个名为 avd1，版本号为 8 的模拟器，输入命令为：

```
android create avd -- name avd1 -- target 8
```

注意，这里的 android 命令是一个批处理文件，它存放在 SDK 所在文件夹的 tools 子文件夹中。如果没有将这个文件夹路径加入到 Windows 操作系统的环境变量 path 中，每次运行 android 命令都必须先进入 tools 子文件夹中，然后才能正常运行。如果不想这么麻烦，建议将 tools 子文件夹的路径添加到环境变量 path 中。另外，还有些 SDK 中的命令是存放在 SDK 所在文件夹的 platform-tools 子文件夹中，例如 adb，所以也建议将 platform-tools 子文件夹的路径添加到环境变量 path 中。本书中 SDK 所在文件夹是 E:\android-sdk，以 Windows 7 为例，向环境变量 path 添加路径的操作如下。

① 打开"计算机"，选择工具栏中的"系统属性"，进入"控制面板"→"系统和安全"→"系统"窗口，选择左侧的"高级系统设置"，进入"系统属性"页的"高级"选项卡。

② 单击“环境变量...”按钮，进入“环境变量”对话框，如图 1-26 所示。

③ 在“系统变量”列表框中选择 Path 变量，然后单击“编辑...”按钮，进入“编辑系统变量”对话框，在“变量值”输入框内加入：“;E:\android-sdk\tools; E:\android-sdk\platform-tools”，在这里每一个路径之间用“;”号分隔。然后单击“确定”按钮，如图 1-27 所示。



图 1-26 “环境变量”对话框



图 1-27 “编辑系统变量”对话框

这样就为系统环境变量 Path 添加了两个子文件夹的路径。在此之后，就可以在命令行状态下的任何目录中运行包含在这两个路径中的命令了。

(2) 在 Eclipse 中创建 AVD 的步骤如下。

① 在 Eclipse 中，选择菜单 Window→AVD Manager，进入 Android Virtual Device Manager 对话框。注意，初始创建时，是没有任何模拟器在列表中的。

② 单击 New...按钮，进入 Create new Android Virtual Device(AVD)对话框开始创建一个新的模拟器，如图 1-28 所示。

在此对话框中，输入 AVD 的名字，在这里为它取名为“AVD_and-23”。在 Target 下拉框中选择 Android 2.3.3 -API Level 10。在 SD Card 中的 Size 输入框内输入“1024”，单位为 MiB，说明这个 SD Card 的存储容量为 1GB。在 Skin 的 Built-in 下拉列表框中选择 HVGA，Built-in 中的选项决定了模拟器屏幕的尺寸，HVGA 代表屏幕的大小是 320×480 像素。

③ 单击 Create AVD 按钮，开始创建模拟器。

(3) 在 Eclipse 中启动 AVD。

① 在 Eclipse 中，选择菜单 Window→AVD Manager，进入 Android Virtual Device Manager 对话框。在模拟器的列表中可以看到已创建的 AVD。

② 选择需要启动的 AVD 项，这里选择的是 AVD_and-23 模拟器，然后单击 Start...按钮，进入 Launch Options 对话框，如图 1-29 所示。

③ 单击 Launch 按钮，就可以启动所选择的模拟器了。注意，首次启动模拟器的时间较长，如果想节省时间，建议在刚一进入 Eclipse 的时候就启动模拟器。启动完成后的模拟器如图 1-30 所示。

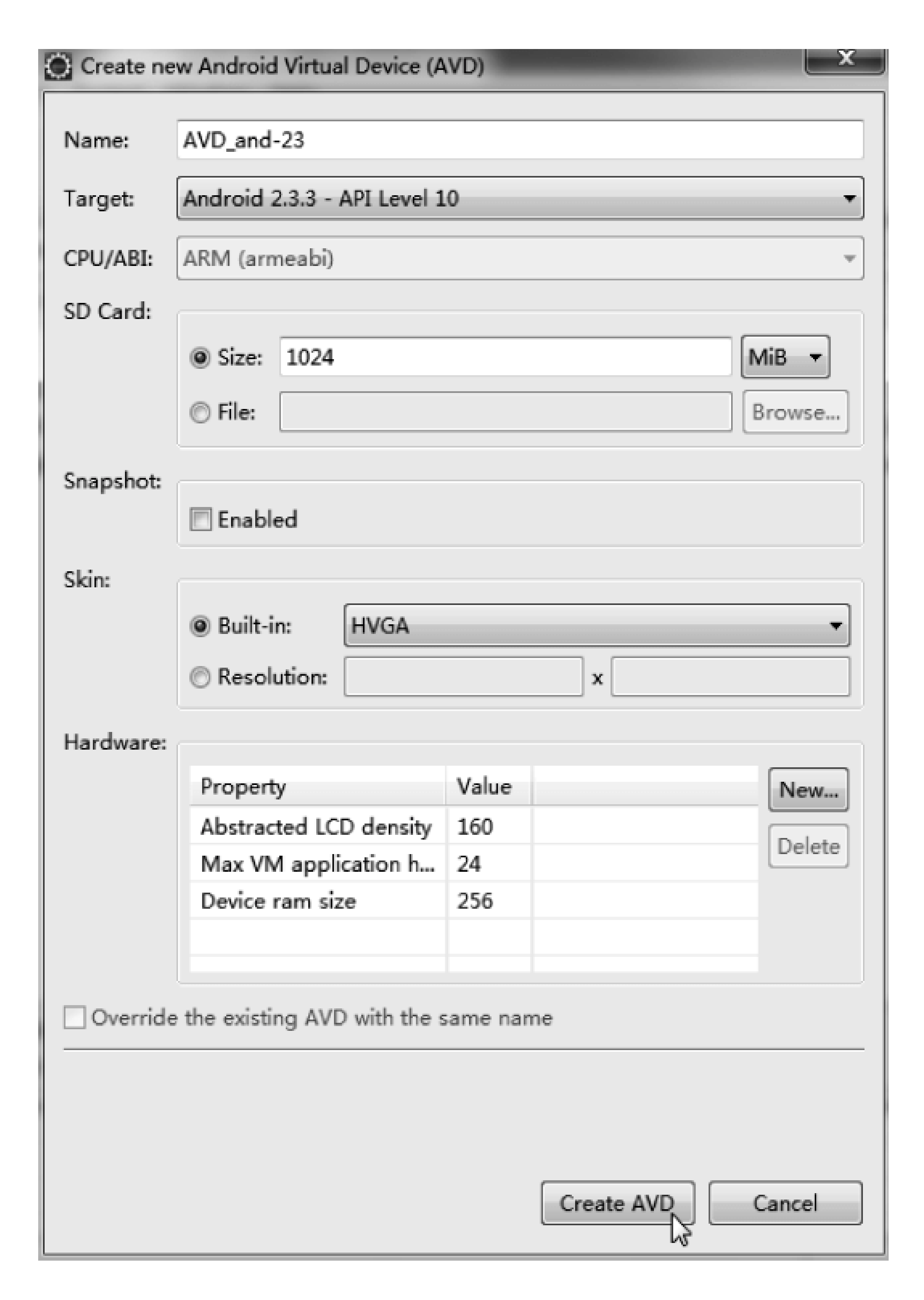


图 1-28 创建 AVD

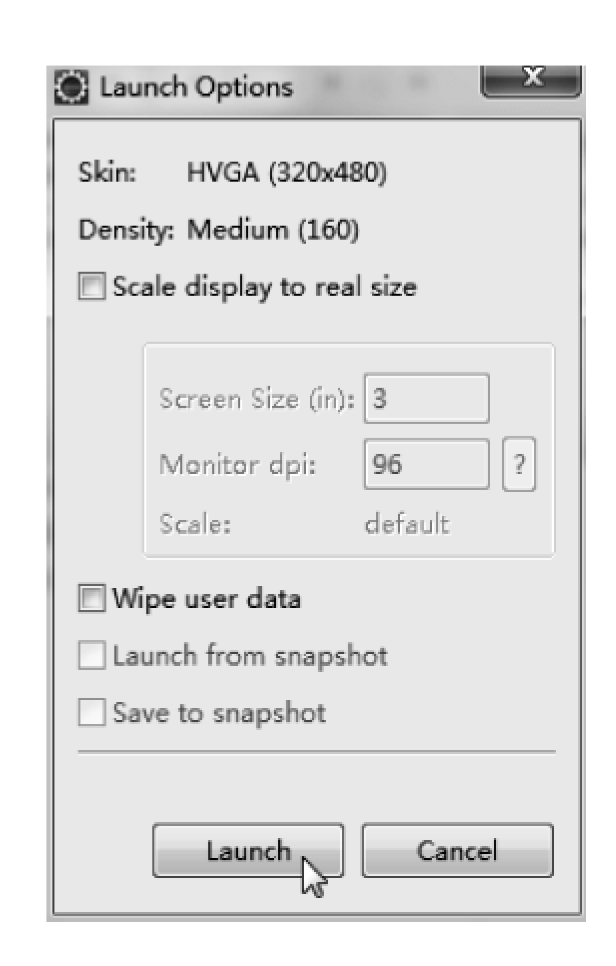


图 1-29 Launch Options 对话框



图 1-30 Android SDK 2.3.3 的模拟器

1.4 Android 的第一个应用

在完成 Android 开发环境的搭建后，就可以开始进入 Android 的应用开发之旅了。本节主要介绍 Android 应用程序的创建和运行。

1.4.1 创建一个 Android 应用项目

在 Eclipse 开发环境下创建 Android 应用项目的步骤如下。

(1) 打开 Eclipse，选择菜单 File→New→Project...，进入 New Project 对话框，选择 Android 下的 Android Project，如图 1-31 所示。

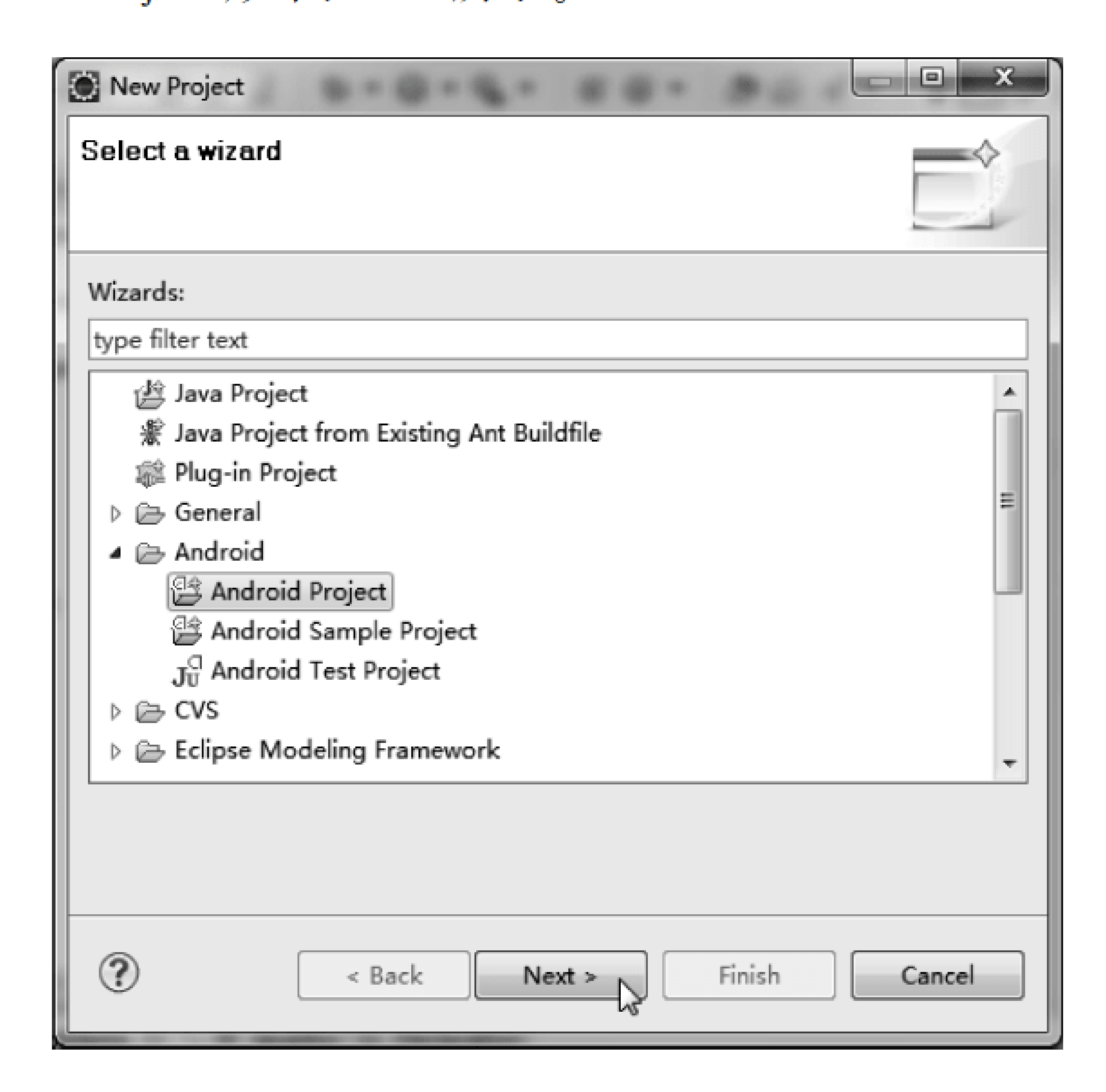


图 1-31 New Project 对话框

(2) 单击 Next 按钮，进入 New Android Project 对话框。在此对话框中输入 Android 应用项目的项目名称。Android 的应用项目名称遵从文件夹命名规则，但不能使用中文名。这里为这个应用项目起名为 HelloAndroid，如图 1-32 所示。

(3) 单击 Next 按钮，进入新建项目选择 SDK 版本页，如图 1-33 所示。

(4) 单击 Next 按钮，进入设置项目参数页，如图 1-34 所示。

在该页中的 Application Name 文本框中输入应用程序名“Hello，Android”。在 Package Name 文本框中输入包名，注意，这个包名必须全球唯一，因为当应用项目发布时，有许多标识是通过包名来区分的，当然，同一个大的项目或同一个开发人员可以为自己的项目定义相同的包名，但前提是不能有相同的项目名或项目中的文件名。在 Create Activity 文本框中输入初始的活动类名“Hello”，这个类名与一个 Java 代码文件名一致，当项目运行时，首先运行的是这个类。注意，这个活动类名不能包含空格。最后是在 Minimum SDK 文本框中输入最小版本号，输入最小版本号是为了定义项目的兼容性，也意味着新建项目能在 SDK 的哪个版本以上正常运行。

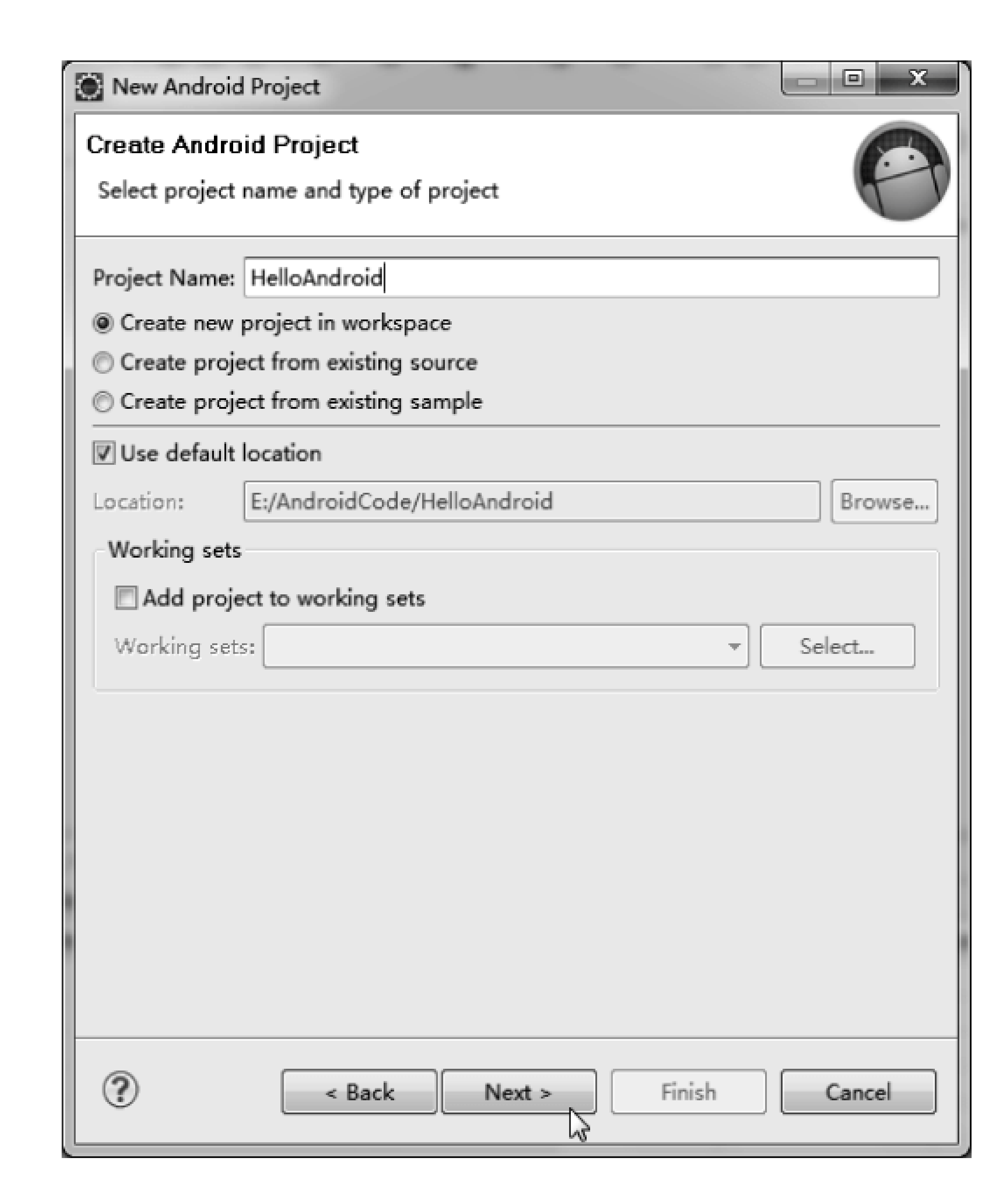


图 1-32　为应用项目命名

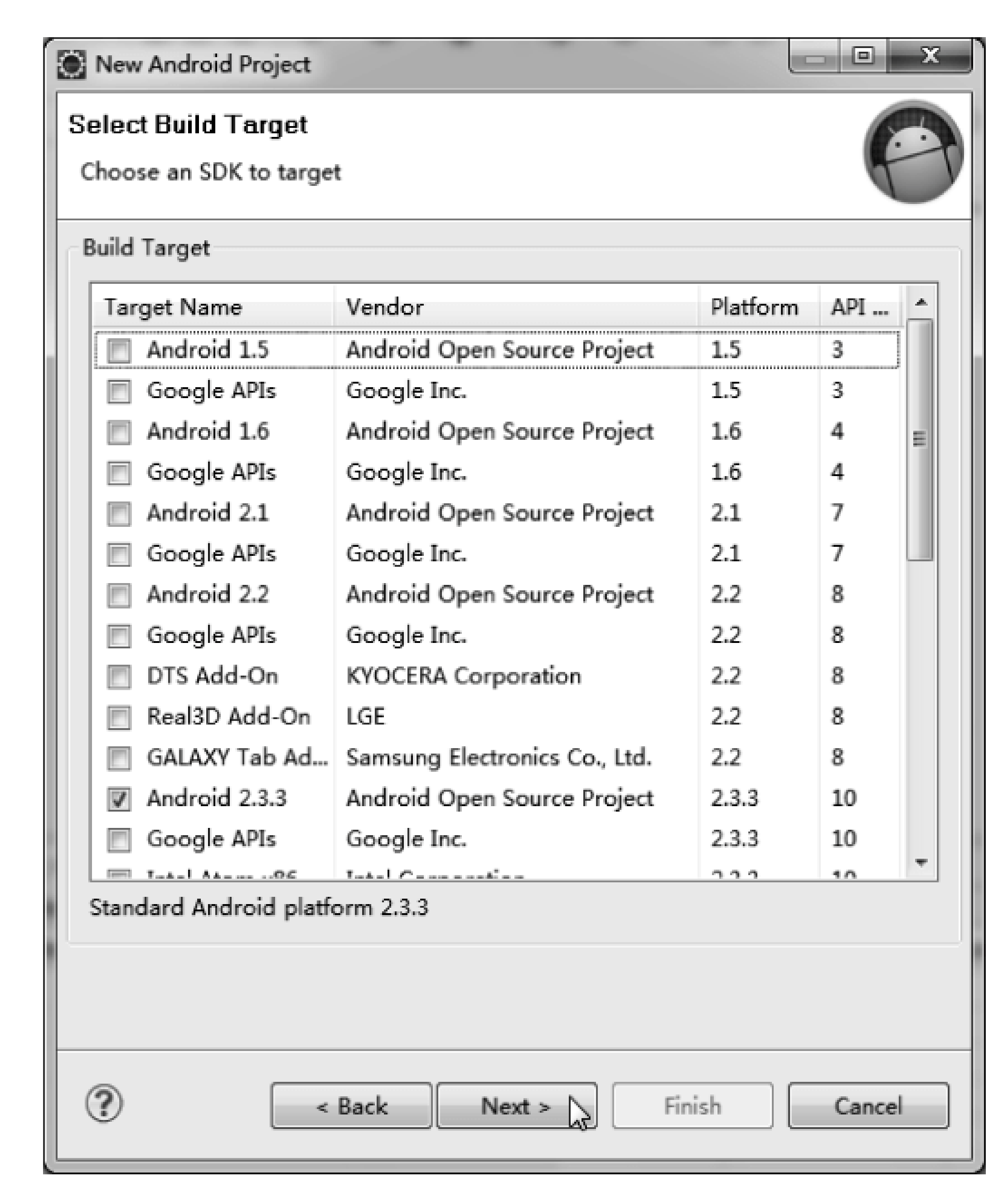


图 1-33　选择新建项目的 SDK 版本页

输入完这个信息后，单击 Finish 按钮，一个 Android 项目就创建完成了。展开这个项目，如图 1-35 所示。

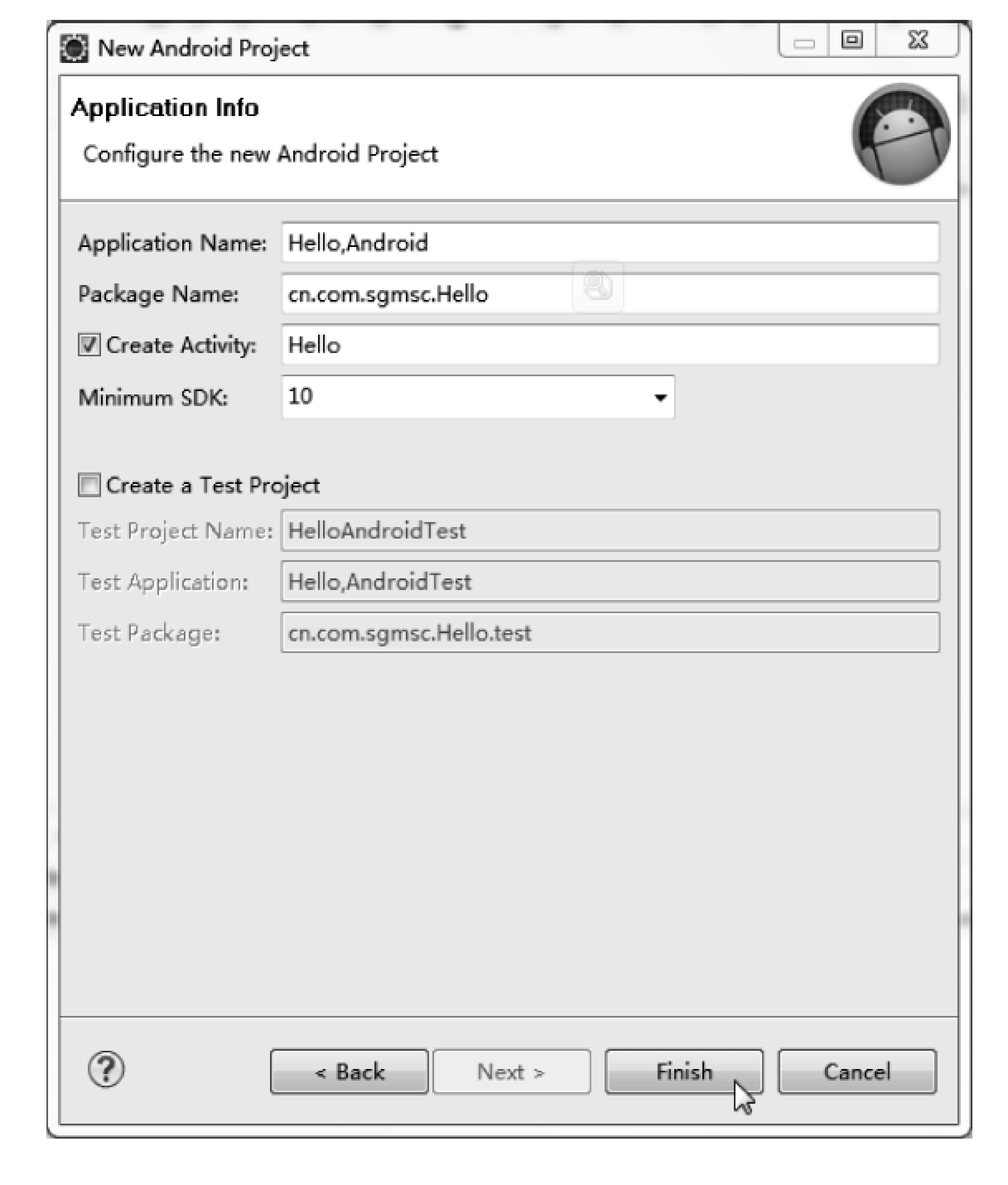


图 1-34　新建应用项目的信息页



图 1-35　HelloAndroid 项目目录

1.4.2 运行 Android 的第一个应用

启动模拟器，在 Eclipse 的 Package Explore 中选择项目名 HelloAndroid，然后单击鼠标右键，在弹出的菜单中选择 Run As→Android Application，即可在模拟器中运行该项目了。其显示结果如图 1-36 所示。

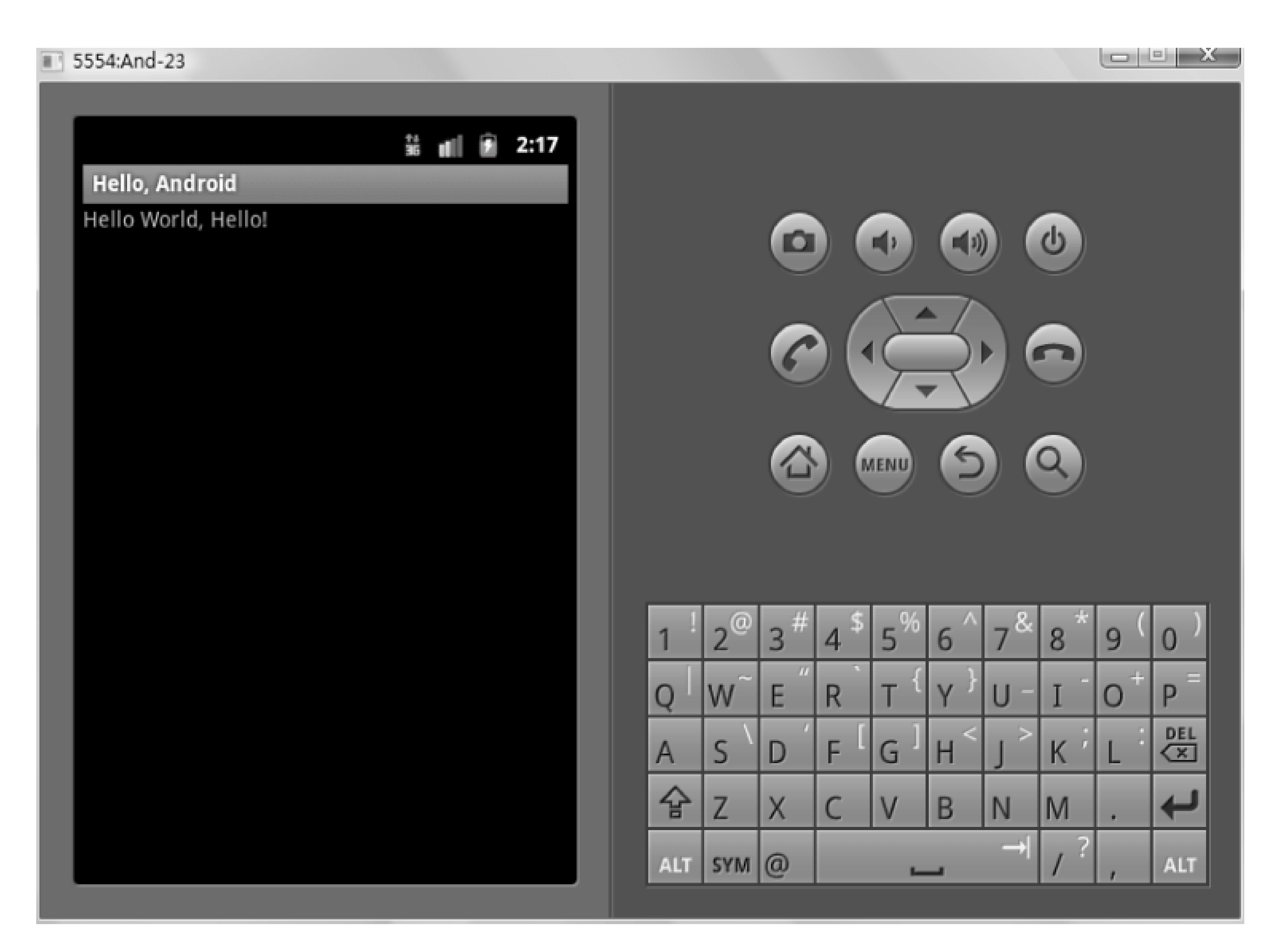


图 1-36 运行 HelloAndroid 项目的效果

如果想横向显示运行结果，可以在 PC 的键盘上使用 Ctrl+F11 组合键，改变模拟器显示方向，如图 1-37 所示。



图 1-37 横向显示 HelloAndroid 项目的运行结果

小结

本章是学习 Android 应用开发的起步，以概述的形式介绍了 Android 平台的诞生背景、发展历程、应用前景，介绍了 Android 平台的框架结构，了解这些知识有助于读者明白学习 Android 应用开发的目的。在本章的后两节中详细地介绍了 Android 开发环境的搭建，Android 项目的创建与运行。相信读者在学习了本章之后，已经完全地掌握了在 Eclipse 环境下开发 Android 应用项目的步骤。

不可避免地，有些读者会问：为什么 HelloAndroid 应用项目没有进行任何编程就可以运行出结果？如果想知道，请学习第 2 章。

练习

1. 独立完成 Android 的环境搭建。
2. 创建并运行一个 HelloAndroid 应用项目。

第2章 Android应用程序的构成

第1章中已经介绍了创建HelloAndroid应用程序的全过程。在创建过程中，没有经过任何代码编程，只是在创建向导中设置了几个名称和选择了一些选项，就可以运行了。在Android中创建一个项目就是这么简单，但是要根据实际应用开发某个项目，还有许多功课要做。

现在，几乎每一个平台都会有自己的结构框架，例如在最初学习Java或者C\C++时，第一个程序总是main函数，以及文件类型和存储方式等。本章将对Android平台的目录结构、文件类型及其各自负责的功能进行剖析。

2.1 Android应用程序目录结构

第1章中建立的HelloAndroid应用项目，其代码是由ADT插件自动生成的，形成Android项目特有的结构框架。在Eclipse中，展开Package Explorer导航器中的HelloAndroid项目，可看到其目录结构，如图2-1所示。

看到这个项目目录，可能会有似曾相识的感觉。其实Android应用程序就是由Java代码和XML属性声明共同设计完成，在编写代码时要注意，Java代码和XML代码都是大小写敏感的。每个Android应用项目都以一个项目目录的形式来组织。下面对其目录结构进行详细的介绍。

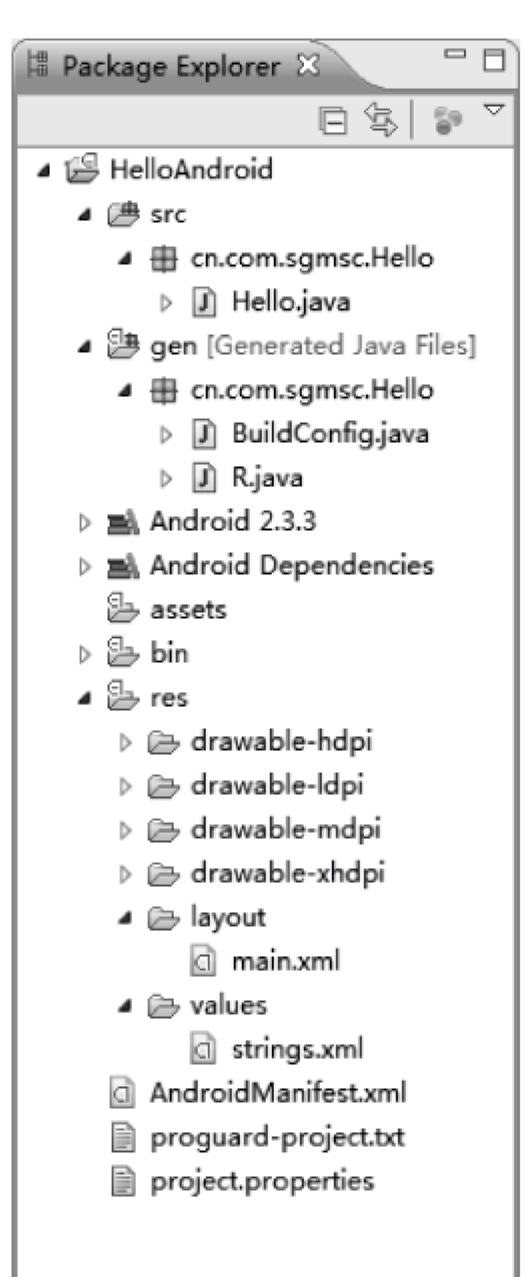


图2-1 HelloAndroid项目的目录

1. src目录

src目录存放Android应用程序中的所有Java源代码，并且以用户所声明的包进行自动地组织。例如在创建HelloAndroid项目时，定义的包为“cn. com. sgmsc. Hello”，那么Hello. java文件就在这个包内，其目录组织方式为src/ cn/com/sgmsc/ Hello/Hello. java。程序员在项目开发过程中，大部分时间是对该目录下的源代码文件进行编写。如果是复杂一点的应用，该目录下会有多个源代码文件。

2. gen目录

gen目录下的文件是由ADT自动生成的，不需要人为去修改。

该目录下只有一个定义在包内的 R.java 文件。R.java 对于 Android 系统非常有用,这个文件中为 Android 项目中的各个资源在相应的类中创建其唯一的 ID,当项目使用到这些资源时,会通过该类得到资源的引用。如果项目资源发生了变化,R.java 都会重新编译,同步更新。

3. Android2.3.3 目录

Android2.3.3 目录存放该项目支持的 jar 包。

4. assets 目录

assets 目录存放项目中用到的相关资源文件,如文本文件、视频文件、MP3 等媒体文件。在程序中可以使用"getResources.getAssets().open("<文件名>")"方法得到资源文件的输入流 InputStream 对象。

5. res 目录

res 目录存放整个项目经常使用的资源文件,该目录称为资源目录,包括一些图标、声音、布局文件及参数描述文件等。当新建一个项目时,系统在该目录中自动建立以下目录。

(1) 以 drawable 开头的 4 个目录:drawable-ldpi、drawable-mdpi、drawable-hdpi 和 drawable-xhdpi 目录,分别存储低、中、高和超高分辨率的图片文件。dpi 是 dot per inch 的缩写,表示每英寸像素数。可使用的图片文件类型如.png、.9.png、.jpg 等图片资源。程序运行时可以根据手机分辨率的高低分别选取相应目录下的图片文件。如果不想准备太多分辨率的图片,也可以只准备一组图片存放于这 4 个目录的任何一个中。

(2) layout 目录:存放应用程序的布局文件,文件类型为 XML 格式。新建项目时 ADT 都会自动创建一个 main.xml 文件。与在网页布局中使用 HTML 文件一样,Android 在 XML 文件中使用 XML 元素来设定屏幕的布局。在 layout 目录中的每个 XML 文件包含整个屏幕或部分屏幕的视图资源。

(3) values 目录:存放所有 XML 格式的资源描述文件,例如字符串(strings.xml)、颜色(colors.xml)、样式(styles.xml)、尺寸(dimens.xml)和数组(arrays.xml)等描述文件。

需要注意的是:res 目录中的所有文件名只能是以"a"~"z"、"0"~"9"或"_"字符命名,不能包含大写字母,否则会导致错误。

在 Java 代码中,可以使用 getResources().getDrawable(resourceId)方法来获取一个图片;可以使用 getResources().getString(resourceId)或 getResources().getText(resourceId)方法来获取 strings.xml 中的一个字符串;可以使用 getResources().get StringArray(resourceId)方法获取一个 String 数组;可以使用 getResources().getColor(resourceId)方法来获取一种颜色等。

问一下:

为什么要有资源描述文件?

有两个好处。以字符串描述文件 strings.xml 为例。一是可以方便地实现双语显示,例如中文和英文。可以将应用程序中要显示的文字以两种语言的形式都存放在 strings.xml 文件中,然后在程序中设置,如果手机在中国地区使用,就显示中文,如果手机在英文国度使用,就显示英文。二是方便程序的维护。如果某些文字在程序中的多处出现,可以将它写入 strings.xml 文件中,而在程序中只是引用它的 ID,这时,如果要修改其文字内容,只需要在 strings.xml 中修改其字符串内容,而不必修改源代码。

6. AndroidManifest.xml 文件

AndroidManifest.xml 文件是项目的系统控制文件，或称为功能清单文件。每个 Android 项目必须有这个文件，位于项目的根目录下。这个清单文件给 Android 系统提供关于这个应用程序的基本信息，系统在运行任何程序代码之前必须知道这些信息。其详细内容将在 2.4 节中进行介绍。

7. proguard-project.txt 文件

proguard-project.txt 文件是混淆代码的脚本配置文件。当创建一个 Android 项目的时候，在项目的根目录下会自动生成 proguard-project.txt 文件。这个文件定义了 proguard 如何优化和混乱项目代码，以得到一个体积更小的并且更难以使用逆向项目(反编译)的 apk 文件，达到一定的加密效果。例如需要对应用添加许可的时候，可以使用 Proguard 工具，使应用难以使用逆向项目。

在 Android 2.3 版本以后，Google 才开始封装代码混淆的脚本文件。Proguard 已经集成在 Android 编译系统之中，所以不必手动去调用它。Proguard 只在以 release 模式编译的时候才会运行，所以当使用 debug 模式编译的时候不必与混乱后的代码打交道。

8. project.properties 文件

project.properties 文件是当前应用所使用 Android 的配置信息。

2.2 Android 应用程序解析

Android 应用程序主要由三部分组成：应用程序描述文件.xml，各种资源，以及应用程序源代码.java。可以这样理解：一个 Android 应用程序，程序员需要做的事是由 Java 代码实现其业务逻辑，由 XML 文件来描述其界面及其他一切资源。

其实，Android 应用的界面设计也可以使用 Java 代码来完成，在实际编程中有时候少数界面组件也是通过代码实现的，这样做比较快捷，但是会增加代码的长度，且不便维护。更多地，界面的设计是通过 XML 文件来描述。这样做的好处在于：界面设计与程序逻辑相分离，使得代码更加短小易维护，更加符合 MVC 设计原则。

问一下：

什么是 MVC 设计？

MVC 是当前流行的设计模式框架，它强制性地将应用程序的输入、处理和输出分开。使用 MVC 应用程序被分成三个核心部件：M 即数据模型，V 即用户界面，C 即控制器，它们各自处理各自的任务。使用 MVC 的目的是将 M 和 V 的代码分离，从而使同一个程序可以使用不同的表现形式。例如一批统计数据可以分别用柱状图、饼图来表示。C 存在的目的则是确保 M 和 V 的同步，一旦 M 改变，V 应该同步更新。

MVC 设计模式的优点在于它的低耦合性、高重用性和可适用性。并且让代码程序员集中精力于业务逻辑开发，让界面程序员集中精力于表现形式上设计，这样大大地降低了开发和维护的技术成本，缩减开发时间，使得应用程序得以快速的部署。但是由于 MVC 没有明确的定义，要完全理解 MVC 并不是很容易，在实际开发中需要花费一些时间去精心计划 MVC 框

架。尽管构造 MVC 应用程序需要一些额外的工作量，但是它带来的好处是毋庸置疑的。

Android 应用程序设计大多是采用 MVC 框架。

下面通过 HelloAndroid 项目的解析来理解 Android 应用程序。

2.2.1 资源及其描述文件

可以从图 2-1 中看到，新建一个 Android 项目，会在 res 下的 drawable-hdpi、drawable-ldpi、drawable-mdpi 和 drawable-xhdpi 目录中分别存放一个默认的图标 icon. png 文件，虽然文件名一样，但它们的分辨率不一样，显示大小不一样。如果在项目中还需用到其他的图片文件，可以将那些图片文件拷贝到上述四个目录中之一。HelloAndroid 项目使用项目创建时的默认图标文件。

1. 资源描述文件

新建一个 Android 项目都会自动创建一个定义字符串常量的描述文件 strings. xml，它存放于 res/values 目录中。

打开 HelloAndroid 项目目录下 res/values 中的 strings. xml 文件，其代码如下。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <resources>
3      <string name = "hello">Hello World, Hello!</string>
4      <string name = "app_name">Hello, Android</string>
5  </resources>
```

(1) 第 1 行声明了 XML 的版本以及编码方式。

(2) <resources>是定义资源的标签。

(3) 第 2～5 行定义两个字符串常量，字符串的名字分别为 hello 和 app_name，其内容分别是“Hello World，Hello!”和“Hello，Android”。定义之后该项目允许在 Java 代码和 XML 代码中使用这些资源文件中的字符串资源。

HelloAndroid 项目只有 strings. xml 资源描述文件。有些应用项目还会有 colors. xml、dimens. xml、styles. xml 等资源描述文件。

1) colors. xml 文件

colors. xml 文件创建布局的颜色常量，使用“<color>”标签定义一个颜色资源。颜色值由 RGB(三位十六进制数)或 RRGGBB(6 位十六进制数)表示，以“#”符号开头。例如：#00f(蓝色)，#00ff00(绿色)。定义透明色，表示透明度的 alpha 通道值紧随“#”之后。例如：#600f(透明蓝色)，#7700ff00(透明绿色)。

2) dimens. xml 文件

dimens. xml 文件通常用于创建布局常量，在样式和布局资源中定义边界、高度和尺寸大小等。使用“<dimen>”标签指定一个维度资源。

常用的维度定义单位如下。

(1) px(像素)：屏幕上的像素。

(2) in(英寸)：长度单位。

(3) mm(毫米)：长度单位。

(4) pt(磅)：1/72 英寸。

(5) dip(与密度无关的像素):一种基于屏幕密度的抽象单位。在每英寸160点的显示器上,1dip=1px。但随着屏幕密度的改变,dip与px的换算会发生改变。

(6) sp(与刻度无关的比例像素):与dip类似,主要处理字体的大小,可以根据用户的字体大小首选项进行缩放。

建议:使用sp作为文字的单位,使用dip作为其他元素的单位。

3) styles.xml文件

styles.xml文件用于预先定义布局中需要显示的样式,如文本的显示颜色和字体等。例如:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <style name="BaseText">
4         <item name="android:textSize">14sp</item>
5         <item name="android:textColor">#111</item>
6     </style>
7     <style name="SmallText" parent="BaseText">
8         <item name="android:textSize">8sp</item>
9     </style>
10 </resources>
```

(1) 第3~6行定义“BaseText”样式,该样式文本的大小为14sp,颜色为#111。

(2) 第7~9行定义“SmallText”样式,它是BaseText的子样式。在保持其父样式的其他属性不变的情况下,定义文本大小为8sp。

2. 界面布局文件

新建的项目会自动创建一个main.xml文件,它定义第一个屏幕界面。一般地,应用程序需要定义多个屏幕界面,程序员根据应用需求,在该目录下进行手动创建。

打开HelloAndroid项目目录下res/layout中的main.xml文件,其代码如下。

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10    android:text="@string/hello"
11    />
12 </LinearLayout>
```

(1) 第1行声明了XML的版本以及编码方式。

(2) 第2~12行向屏幕界面添加一个垂直的线性布局。

(3) 第2行标签中的属性“xmlns:android="http://schemas.android.com/apk/res/android"”是XML命名空间的声明,它告诉Android的工具,你将要涉及公共的属性已被定义在XML命名空间。在每一个Android的布局文件的最外层标签必须有这个属性。

(4) 第7~11行向线性布局中添加一个文本控件TextView,显示的内容为strings.xml文件中常量“Hello”定义的字符串内容。

3. R.java 文件

gen 目录下 cn.com.sgmsc.Hello 包中的 R.java 文件为 Android 项目中的各个资源创建了唯一的 ID。打开 R.java 文件,其代码如下。

```
1  package cn.com.sgmsc.Hello;
2
3  public final class R {
4      public static final class attr {
5      }
6      public static final class drawable {
7          public static final int icon = 0x7f020000;
8      }
9      public static final class layout {
10         public static final int main = 0x7f030000;
11     }
12     public static final class string {
13         public static final int app_name = 0x7f040001;
14         public static final int hello = 0x7f040000;
15     }
16 }
```

(1) 第 1 行是声明 R.java 在该包下。

(2) 第 3~16 行定义 R 类。

(3) 第 6~8 行为项目中使用的图片资源创建 ID。

(4) 第 9~11 行为项目中使用的屏幕布局创建 ID。

(5) 第 12~15 行为项目中使用的字符串常量创建 ID。

就这样,R.java 把项目中需要使用的资源,用 ID 的形式标注了 drawable、layout、values 三个目录中的资源信息,其中 values 目录只有字符串资源文件 strings.xml。

4. 功能清单文件

每个 Android 项目都必须有一个清单文件,用于应用程序的全局描述。当新建一个项目时,ADT 自动创建一个 AndroidManifest.xml 文件,存放在项目的根目录下。

打开 HelloAndroid 项目根目录下的 AndroidManifest.xml 文件,其代码如下。

```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
      package = "cn.com.sgmsc.Hello"
      android:versionCode = "1"
      android:versionName = "1.0">
    <application android:icon = "@drawable/icon" android:label = "@string/app_name">
        <activity android:name = ".Hello"
                  android:label = "@string/app_name">
            <intent-filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion = "10" />
</manifest>
```

（1）第 1 行声明了 XML 的版本以及编码方式。

（2）第 2～16 行声明该应用程序所包含的包名、组件、资源、可兼容的最低版本等信息。

（3）第 2 行标签中的属性“xmlns:android="http://schemas.android.com/apk/res/android"”是 XML 命名空间的声明。在每一个 Android 的清单文件的最外层标签必须有这个属性。

（4）第 3 行声明应用程序的包名。

（5）第 6 行声明应用程序的图标和标签名称，图标使用 res/drawable 目录下的 icon.png 文件，标签名称使用 res/values 目录下的 strings.xml 中常量“app name”定义的字符串内容。

（6）第 7 行声明应用程序第一个运行的 Java 代码是 src 下包中的 Hello.java 文件。

（7）第 8 行声明屏幕上的标题显示内容是 strings.xml 中常量“app name”定义的字符串。

（8）第 15 行声明应用程序可兼容的最低版本号为 10。

AndroidManifest.xml 文件中的这些声明大多都是在创建 HelloAndroid 项目时定义的，如图 2-2 所示。

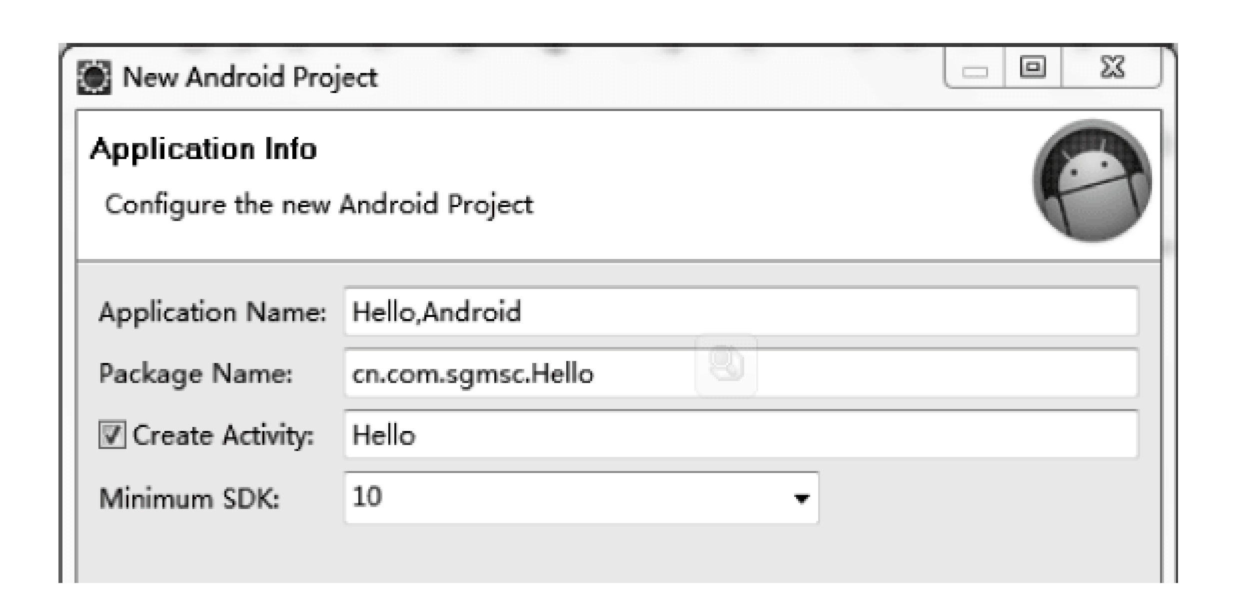


图 2-2 创建 HelloAndroid 项目的定义项

2.2.2 逻辑代码文件

应用程序的操作控制部分在 Java 源程序中定义。创建 HelloAndroid 项目时，在图 2-2 的 Create Activity 框中输入的“Hello”，就是定义该项目的逻辑代码文件名为“Hello.java”。

Hello.java 位于项目的 src 目录中。打开 HelloAndroid 项目下 src/cn.com.sgmsc.Hello 中的 Hello.java 文件，其代码如下。

```
package cn.com.sgmsc.Hello;

import android.app.Activity;
import android.os.Bundle;

public class Hello extends Activity {
    //当第一次创建该 Activity 时回调该方法
    @Override
    public void onCreate(Bundle savedInstanceState) {
        //调用父类的 onCreate 构造函数，savedInstanceState 是保存当前 Activity 的状态信息
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                //设置当前显示的布局
     }
}
```

(1) 第 1 行是声明 Hello.java 在该包下。

(2) 第 3～4 行是引入相关的类。

(3) 第 6 行是定义 Hello 类的开始,它表示 Hello 是一个公开的继承自 Activity 的类,而这个 Activity 就是第 3 行引入的类。其全部代码写在一对"{}"大括号内。注意,在 Java 代码中每个类名都必须与其相应的代码文件名一致。

(4) 第 6～14 行定义 Hello 类。

(5) 第 8 行的@Override 表示要重写紧跟在它下面的方法。它下面的是 onCreate()方法,此时是重写 onCreate()方法。因为 Hello 类是继承自 android.app.Activity 类的,原来的 Activity 类有定义 onCreate()方法,重写 onCreate()方法就是重新定义 Hello 类自己的 onCreate()方法。当应用程序执行时,就会执行 Hello 类自己的 onCreate()方法里的代码。

(6) 第 9～13 行重新定义了 onCreate()方法,其中 onCreate()方法传入了一个名为 savedInstanceState 的 Bundle 类型参数。

(7) 第 11 行是调用 Hello 类的父类即 Activity 类的 onCreate()方法。几乎每一个 Android 代码程序在重写 onCreate()方法时,都会先调用其父类的 onCreate()构造方法。Bundle 类型参数 savedInstanceState 用于保存当前 Activity 的状态信息。

(8) 第 12 行设置当前显示的布局,即显示 res/layout/main.xml 文件中定义的屏幕布局。

回顾一下,strings.xml 文件定义了字符串常量,main.xml 文件定义了显示文本,AndroidManifest.xml 声明了应用项目的显示图标和名字、屏幕标题显示的文本,并指出代码文件是 cn.com.sgmsc.Hello 包中的 Hello.java 文件,Hello.java 文件指定屏幕按 res/layout/main.xml 文件中定义的布局显示。因此,HelloAndroid 项目运行的结果如图 2-3 所示。

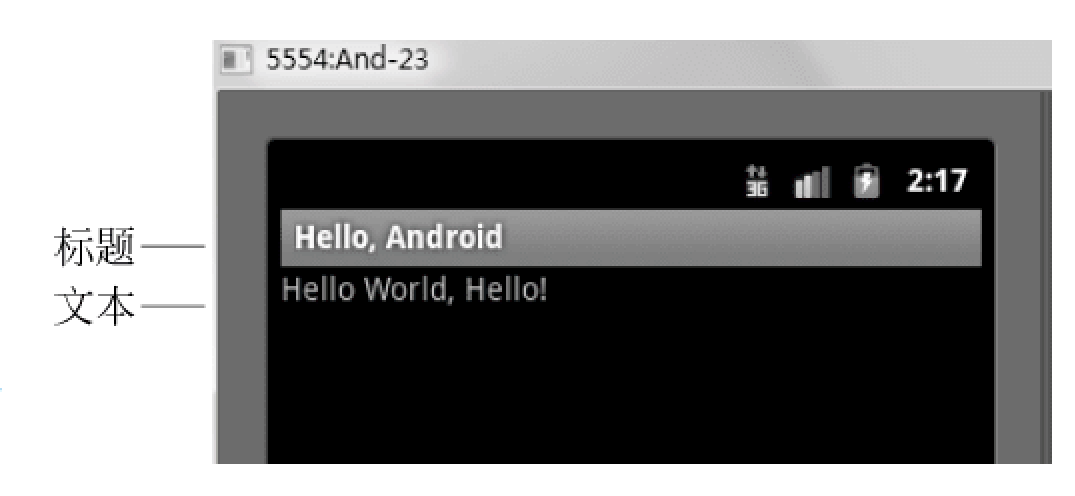


图 2-3 HelloAndroid 运行的结果

2.3 Android 的基本组件

Android 是一个为组件化而搭建的平台,它的应用程序由一些零散的有联系的组件组成,并通过工程的 AndroidManifest.xml 把它们绑定在一起。

Android 应用程序常用的组件有 Activity(活动)、Service(服务)、Broadcast Receiver(广播接收器)、Intent(意图)、Content Provider(内容提供器)和 Notification(通知)等。然而,并不是所有的应用程序都必须包含这些组件,但一般是由上面的一个或多个组件来组建。当你决定使用以上哪些组件来构建 Android 应用程序时,就应该将它们列在 AndroidManifest.xml 文件中,并且声明它们的特性和要求。

1. Activity

Activity 是 Android 中最常用的组件,一个 Activity 展现一个可视化的用户界面,它是应用程序的显示层。例如,一个 Activity 可能展现为一个用户可以选择的菜单项列表,或者展现一些图片以及图片的标题。通常一个应用程序会有多个 Activity。虽然这些 Activity 一起工作,但每一个 Activity 都是相对独立的。每个 Activity 都继承自 android.app.Activity 类。

Activity 显示的每一个内容都是由 View（视图）对象去构建，并定义在 res/layout 下的 XML 文件中。Android 自带了很多 View 对象，例如按钮、文本框、滚动条、菜单、多选框等。每个视图或视图组对象在布局文件中都有它们自己的 XML 属性，其中的 ID 属性唯一来标识这个视图对象，这个 ID 属性有时被定义为字符串，编译后为整型数值。

如果在 HelloAndroid 的 main.xml 中的 TextView 对象增加一个 ID 属性如下，见下列代码的第 8 行。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:orientation = "vertical"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      >
7  < TextView
8      android:id = "@ + id/tvStr"
9      android:layout_width = "fill_parent"
10     android:layout_height = "wrap_content"
11     android:text = "@string/hello"
12     />
13 </LinearLayout >
```

再打开 R.java，可看到增加了一个 ID，见下列代码的第 9～11 行。

```
1  package Hello.cn.com.seles.www;
2
3  public final class R {
4      public static final class attr {
5      }
6      public static final class drawable {
7          public static final int icon = 0x7f020000;
8      }
9      public static final class id {
10         public static final int tvStr = 0x7f050000;
11     }
12     public static final class layout {
13         public static final int main = 0x7f030000;
14     }
15     public static final class string {
16         public static final int app_name = 0x7f040001;
17         public static final int hello = 0x7f040000;
18     }
19 }
```

问一下：

“@＋id”与“@id”的区别？

我们经常在布局文件中看到 View 对象的 ID 属性声明，有时候用“@＋id”，有时候用“@id”。“@＋id”表示在 R 文件中产生一个新的 ID，如果没有“＋”，而是“@id”，就是引用其他地方已经定义过的 ID，例如“android:id＝"@＋id/tvStr"”。这里，“@”符号是将这个 View 对象的 ID 自动记载在 R 文件中，“＋”号表示向 R 文件中的内部类 ID 中添加一个变量，名字叫 tvStr。

启动一个 Activity 有三种方法。一般在 onCreate()方法内调用 setContentView()方法，

用来指定将要启动的 res/layout 目录下的布局文件，例如 setContentView(R. layout. main)。第二种方法是调用 startActivity()，用于启动一个新的 Activity。第三种方法是调用 startActivityforResult()，用于启动一个 Activity，并在该 Activity 结束时会返回信息。

在 HelloAndroid 应用程序中，使用的就是 setContentView()方法来启动这个 Activity 的。其 Hello. java 代码如下。

```
package cn.com.sgmsc.Hello;

import android.app.Activity;
import android.os.Bundle;

public class Hello extends Activity {
    //当第一次创建该 Activity 时回调该方法
    @Override                                              //重写 onCreate()方法
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                     //设置当前显示的布局
     }
}
```

返回一个 Activity 也有三种方法。通常调用 finish()方法来关闭一个 Activity。如果调用 setResult()方法，则可以返回数据给上一级的 Activity。当使用 startActivityforResult()启动的 Activity 时，则需要调用 finishActivity()方法，来关闭其父 Activity。

2. Service

Service 没有用户界面，但它会在后台一直运行。例如，Service 可能在用户处理其他事情的时候播放背景音乐，或者从网络上获取数据，或者执行一些运算，并把运算结果提供给 Activity 展示给用户。每个 Service 都继承自 Serivce 类。

Service 一般由 Activity 启动，但是并不依赖于 Activity。Service 具有较长的生命周期，即使启动它的 Activity 的生命周期结束了，Service 仍然会继续运行，直到自己的生命周期结束为止。

Service 的启动方式有两种，startService 方式和 bindService 方式。

(1) 使用 startService 方式启动。当在 Activity 中调用 startService()方法启动 Service 时，会依次调用 onCreate()和 onStart()方法；调用 stopService()方法结束 Service 时，会调用 onDestroy()方法。

(2) 使用 bindService 方式启动。调用 bindService()方法启动 Service 时，会依次调用 onCreate()和 onBind()方法；调用 unbindService()方法结束 Service 时，会调用 onUnbind()和 onDestroy()方法。

3. Broadcast Receiver

Broadcase Receiver 不执行任何任务，仅是接受并响应广播通知的一类组件。大部分广播通知是由系统产生的，例如改变时区，电池电量低，用户选择了一幅图片或者用户改变了语言首选项。应用程序同样也可以发送广播通知，例如通知其他应用程序某些数据已经被下载到设备上可以使用。一个应用程序可以包含任意数量的 Boradcase Reveiver 来响应它认为很重要的通知。每个 Broadcast Receiver 都继承自 BroadcastReceiver 类。

Broadcast Receiver 不包含任何用户界面。当系统或某个应用程序发送了广播时，可以使用 BroadcastReceiver 组件来接收广播消息，并做出相应的处理，如：闪动背景灯、振动设备、发出声音等。通常程序会在状态栏上放置一个持久的图标，用户可以打开这个图标并读取通知信息。

BroadcastReceiver 的使用过程如下。

(1) 将信息封装，并添加到一个 Intent 对象中。

(2) 然后通过调用 Context. sendBroadcast ( )、Context. sendOrderedBroadcast ( ) 或 Context. sendStickyBroadcast()方法将 Intent 对象广播出去。

(3) 接收者检查注册的 IntentFilter 是否与收到的 Intent 相同。

(4) 如果相同便会调用 onReceive()方法来接收消息。

BroadcastReceiver 的三个发送方法的不同之处是：使用 sendBroadcast()或 sendStickyBroadcast()方法时，所有满足条件的接收者都会随机地执行，而使用 sendOrderedBroadcast()方法时，接收者会根据 IntentFilter 中设置的优先级顺序地来执行。

需要注意的是，使用 BroadcastReceiver 需要先注册。注册 BroadcastReceiver 对象的方式有两种，一种是在 AndroidManifest. xml 中声明，另一种是在 Java 代码中设置。

(1) 在 AndroidManifest. xml 中声明时，要将注册的信息放在<receiver> </receiver>标签之中，并通过<intent-filter>标签来设置过滤条件。

(2) 在 Java 代码中设置时，需要先创建 IntentFilter 对象，并在 IntentFilter 对象内设置 Intent 过滤条件，再通过调用 Context. registerReceiver()方法来注册监听，然后通过 Context. unregisterReceiver()方法来取消监听。用此方式的缺点是当 Context 对象被销毁时，该 BroadcastReceiver 对象也就随之被销毁了。

4. Intent

1) Intent 概述

Intent 就是连接各组件的桥梁，是一种运行时的绑定机制，Android 提供了 Intent 机制来协助应用间的交互与通信。Intent 负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述，Android 则根据此 Intent 的描述，负责找到对应的组件，将 Intent 传递给调用的组件，并完成组件的调用。Intent 不仅可用于应用程序内部的 Activity、Service 和 BroadcastReceiver 之间的交互，也可用于应用程序之间的交互。因此，可以将 Intent 理解为不同组件之间通信的“媒介”，专门提供组件互相调用的相关信息。

前面介绍的 Activity、Service 和 BroadcastReceiver 组件之间的通信全部使用的是 Intent，但是各个组件使用的 Intent 机制不同。

(1) 当需要激活 Activity 组件时，调用 Context. startActivity()或 Context. startActivityForResult()方法来传递 Intent。

(2) 当需要激活 Service 组件时，调用 Context. startService ()或 Context. bind Service ()方法来传递 Intent。

(3) 当需要使用 BroadcastReceiver 组件时，调用 sendBroadcast()、sendStickyBroadcast()或 sendOrderedBroadcast()方法来传递 Intent，当 BroadcastReceiver 被广播后，所有 IntentFilter 过滤条件满足的组件都将被激活。

Intent 是由组件名称、Action、Data、Category、Extra 及 Flag 6 部分组成的，在第 3 章中将

对其进行详细的介绍。

2) IntentFilter

IntentFilter(Intent 过滤器)正如其英文的意思,是 Intent 过滤器。一个应用程序开发完成后,需要告诉 Android 系统自己能够处理哪些隐性的 Intent 请求,这就需要声明 IntentFilter。通常在 AndroidManifest. xml 文件中声明。IntentFilter 的使用方法实际上非常简单,只须在 AndroidManifest. xml 中的<Intent-filter>标签指定该应用程序能接收的 Intent 值即可,在一个<Intent-filter>标签中可以设置多个过滤值。

5. Content Provider

Content Provider 用来管理和共享应用程序的数据存储,是 Android 提供的一种标准的共享数据的机制。在应用程序间,Content Provider 是共享数据的首选方式。这意味着,用户可以配置自己的 Content Provider 去存取其他的应用程序或者通过其他应用程序暴露的 Content Provider 去存取它们的数据。Content Provider 都继承自 ContentProvider 类。

Content Provider 创建其他程序使用的数据集。数据可以存在系统的 SQLite 数据库或者其他地方。ContentProvider 通过实现一组标准的方法,来使其他程序可以存取数据。但是,程序并不是直接调用这些方法,而是使用 ContentResolver 对象来调用这些方法。对于 ContentProvider 而言,最重要的就是数据模型(Data Model)和 URI。

(1) 数据模型(Data Model)。ContentProvider 为所有需要共享的数据创建一个数据表,在表中,每一行表示一条记录,而每一列代表某个数据,并且其中每一条数据记录都包含一个名为"_ID"的字段类标识每条数据。

(2) URI(Uniform Resource Identifier,通用资源标识符)。每个 ContentProvider 都会对外提供一个公开的 URI 来标识自己的数据集。URI 主要分为三个部分:scheme、authority 和 path,其中 authority 又分为 host 和 port。其格式为"scheme://host:port/path"。在 Android 中,所有的 URI 都以"content://"开头,例如:"content://com. example. project:200/folder/subfolder/etc"。

需要注意的是,使用 ContentProvider 访问共享资源时,要为应用程序添加适当的权限。在应用程序的 AndroidManifest. xml 文件中添加权限"<uses-permission android:name="android. permission. READ_CONTACTS"/>"。

6. Notification

Notification 用来在不需要焦点或不中断它们当前 Activity 的情况下提示用户。它们是 Service 或 Broadcast Receiver 获得用户注意的首选方式。例如,当设备收到文本信息或外部来电时,它通过闪光、发声、显示图标或显示对话框信息来提醒用户。

2.4 AndroidManifest. xml 文件

从本章的前面叙述中,可以体会到应用程序的功能清单文件 AndroidManifest. xml 非常重要,每一个应用程序不可缺少。下面对该文件进行详细介绍。

2.4.1 AndroidManifest.xml 的主要构成

AndroidManifest.xml 是应用程序的全局描述文件，它让外界知道该应用程序包含哪些组件、哪些资源及何时运行该程序等信息。其主要内容如下。

✍应用程序的包名，该包名将作为应用程序的唯一标识符。

✍包含的组件如：Activity、Service、BroadcastReceiver 及 ContentProvider 等。

✍应用程序兼容的最低版本。

✍声明应用程序需要的链接库。

✍声明应用程序自身应该具有的权限。

✍其他应用程序访问该应用程序时应该具有的权限。

当新创建一个应用项目时，系统会自动生成 AndroidManifest.xml 文件。但是从 HelloAndroid 项目中可以看到，生成的 AndroidManifest.xml 文件信息不多，真正有实用价值的应用都需要对它进行再编辑。在 Eclipse 的默认情况下，刚一打开项目的 AndroidManifest.xml 文件，会看到一个可视化编辑器，如图 2-4 所示。



图 2-4 AndroidManifest.xml 的可视化编辑器

图 2-4 是 Android 的 ADT 插件提供的可视化编辑器，通过选项的输入和设置可以完成 AndroidManifest.xml 文件。但是作为一位程序员，更多地是选择直接进行录入编辑。即单击该编辑器下方的 AndroidManifest.xml 标签，进入编辑页，如图 2-5 所示。

下面分别讲解 AndroidManifest.xml 文件中的标签。

AndroidManifest.xml 文件是以 XML 格式描述，开头都会出现“<?xml version="1.0" encoding="utf-8"?>”这段叙述，<manifest>标签是根节点，AndroidManifest.xml 的其他标签都定义在该标签下。在此标签的属性中必须要指定“xmlns:android="http://schemas.android.com/apk/res/android"”，并且声明了应用程序所在的包名、版本号。

代码：

```
package = " cn.com.sgmsc.Hello"
```



图 2-5　AndroidManifest.xml 的编辑页

"package"是<manifest>标签的一个特别属性，它声明了应用程序所在的包名，即指定这个应用程序的进入点存在于"cn.com.sgmsc.Hello"这个名称空间路径中。

代码：

```
android:versionCode = "1"
android:versionName = "1.0"
```

"android:versionCode"和"android:versionName"是应用程序版本号。这两个属性是可选的（非必要）。"android:versionName"是给使用者看的版本号，如"1.0"、"2.0"。"android:versionCode"则是开发者用的内部版本号，一般使用流水号。

代码：

```
< uses - sdk android:minSdkVersion = "10" />
```

Android SDK 1.1 版之后引入了这条叙述。透过指定这个参数，系统可以依此辨别应用程序是否使用相容的 SDK 版本，以决定能否在这台机器上安装执行。"1"代表 Android SDK 1.0，"2"代表 SDK 1.1，"3"代表 SDK 1.5，"10"代表 SDK 2.3.3。这也是一个可选填的选项。但如果应用程序要发布出去，一些强势的应用发布平台如 Google Android Market 已规定所有新发布的应用程序必须指定"android:minSdkVersion"这个参数。

在<manifest>标签中可以包含 0 个或 1 个<application>标签。见如下代码。

```
< application android:icon = "@drawable/icon" android:label = "@string/app_name">
…
</application>
```

在<application>标签内，定义所有这个应用程序用到的 Activity、Service 等信息。标签中的"android:icon"属性，定义了这个应用程序将显示在 Android 主画面中的应用程序图标。"android:icon = "@drawable/icon""表示应用程序图标取自资源"res/drawable/icon"；"android:label"属性可用来指定应用程序将显示在 Home 主画面上的名称，也就是预设刚开好机时，可以从桌面下方拉出的应用程序列表项。

在 AndroidManifest. xml 中有时还要声明权限，这部分内容将在 2.4.2 节中说明。下面来讲讲在<application>标签中对应用程序所包含的组件如何进行声明。

在<application>标签中可以包含 0 个或多个 Activity、Service、BroadcastReceiver 及 ContentProvider 等组件的声明。如果已经在程序代码中定义好了组件，例如已定义好一个 Activity，却没有在“AndroidManifest. xml”中加入相应“activity”节点，那么在执行时是无法启动这个 Activity 的。其他的组件也是一样。

1. <activity>标签

Activity 组件使用<activity>标签，例如下列代码：

```
<activity android:name = ".Hello" android:label = "@string/app_name">
  <intent - filter>
      <action android:name = "android.intent.action.MAIN" />
      <category android:name = "android.intent.category.LAUNCHER" />
  </intent - filter>
</activity>
```

<activity>的“android:name”属性，指出了这个 Activity 所对应的类。因为在上一层<manifest>标签属性中已经定义了“package=" cn. com. sgmsc. Hello"”，所以类名就是在此包下的类“. Hello”。“android:label”属性可用来指定应用程序将显示在 Activity 画面上方标题栏的名称。

在<activity>标签中一般需要定义 0 个或多个<intent-filter>声明，用于设置该<activity>的 Intent 过滤条件。<intent-filter>中主要包含两个子标签：“action”和“category”，用于设置其 Intent 过滤条件，来实现与其他组件的通信。<action>标签中的“android:name”属性，其内容“android. intent. action. MAIN”表示：这个 Activity 是此应用程序为进入点（就像 Java 程序中常见的 main 主程序一样），开启这个应用程序时，应先执行这个 Activity。<category>标签中的“android: name”属性，其内容“android. intent. category. LAUNCHER”表示：这个 Activity 将显示在 Launcher 的应用程序列表中。

于是，可以这样理解 HelloAndroid 应用的 AndroidManifest. xml 文件所传达的信息：在“cn. com. sgmsc. Hello”路径下的“Hello. java”这个文件中，已定义了一个主要的 Activity；当打开 Android 的时候，显示的是位于“res/drawable/icon”的图示。一旦按下图示来启动这个应用程序，Android 应用程序框架会去寻找到定义了“android. intent. action. MAIN”内容的“. Hello”Activity 并呼叫执行，它定义在“Hello. java”文件中。

HelloAndriod 是一个非常简单的应用程序，所以在 AndroidManifest. xml 中没有包含其他的组件声明，下面对其他的组件声明作简单介绍。

2. <service>标签

Service 组件使用<service>标签，除了没有屏幕显示外，其他的定义与 Activity 几乎一样，也可以在该节点下包含<intent-filter>等声明，通过 Intent 可实现与其他组件的通信。例如下列代码：

```
<service
   android:enabled = "true" android:name = ".MyService">
</service>
```

3. ＜receiver＞标签

BroadcastReceiver 组件使用＜receiver＞标签，它与＜service＞标签的定义差不多，也可以在该节点下包含＜intent-filter＞等声明，通过 Intent 可实现与其他组件的通信。例如下列代码：

```
<receiver android:enabled = "true"
    android:label = "My Broadcast Receiver"
    android:name = ".MyBroadcastReceiver">
</receiver>
```

4. ＜provider＞标签

ContentProvider 组件使用＜provider＞标签，它通过一组标准的方法接口，以及相应的权限，用来实现数据的共享。例如下列代码：

```
<provider android:permission = "com.paad.MY_PERMISSION"
    android:name = ".MyContentProvider"
    android:enabled = "true"
    android:authorities = "com.paad.myapp.MyContentProvider">
</provider>
```

2.4.2 应用程序的权限

有时在 AndroidManifest.xml 文件中还需指定相应的权限，使得应用程序能够正常工作，或者程序包必须被授予该权限，以限制哪些应用可以访问指定的程序包内的组件和特有功能。例如网络权限、发送短信权限等。

在 AndroidManifest.xml 中声明权限使用＜uses-permission＞标签，例如下列代码：

```
<uses-permission
        android:name = "android.permission. ACCESS_FINE_LOCATION">
</uses-permission>
```

描述权限使用“android.permission.权限常量”格式，权限常量由大写英文单词加下划线“_”组成。表 2-1 中列出部分常用的权限常量。

表 2-1 部分常用的权限常量

权限常量	含义
ACCESS_FINE_LOCATION	允许一个程序访问精良位置(如 GPS)
ACCESS_LOCATION_EXTRA_COMMANDS	允许应用程序访问额外的位置提供命令
ACCESS_WIFI_STATE	允许程序访问 Wi-Fi 网络状态信息
ADD_SYSTEM_SERVICE	允许程序发布系统级服务
BATTERY_STATS	允许程序更新手机电池统计信息
BLUETOOTH	允许程序连接到已配对的蓝牙设备
BLUETOOTH_ADMIN	允许程序发现和配对蓝牙设备
CALL_PHONE	允许一个程序初始化一个电话拨号，不需通过拨号用户界面来确认

续表

权限常量	含义
CAMERA	请求访问使用照相设备
CHANGE_CONFIGURATION	允许一个程序修改当前设置,如本地化
CHANGE_NETWORK_STATE	允许程序改变网络连接状态
CHANGE_WIFI_STATE	允许程序改变 Wi-Fi 连接状态
FLASHLIGHT	允许访问设备上的闪光灯
INTERNET	允许程序打开网络套接字
READ_CALENDAR	允许程序读取用户日历数据
READ_CONTACTS	允许程序读取用户联系人数据
READ_SMS	允许程序读取短信息
REBOOT	请求能够重新启动设备
RECEIVE_SMS	允许程序监控一个将收到短信息,记录或处理
SEND_SMS	允许程序发送 SMS 短信
SET_TIME_ZONE	允许程序设置时间区域
SET_WALLPAPER	允许程序设置壁纸
VIBRATE	允许访问振动设备
WRITE_CALENDAR	允许一个程序写入但不读取用户日历数据
WRITE_CONTACTS	允许程序写入但不读取用户联系人数据
WRITE_GSERVICES	允许程序修改 Google 服务地图
WRITE_OWNER_DATA	允许一个程序写入但不读取所有者数据
WRITE_SMS	允许程序写短信

2.4.3 范例

在下载并安装了 Android SDK 之后,在其安装目录下会有一些范例。在第 1 章的系统环境安装时,SDK 的安装路径为 E:\android-sdk,该文件夹下有个“samples”子文件夹,这里面有一些经典的 Android 应用程序示例,分别以不同的 SDK 版本组织在不同的子文件夹中。打开 E:\android-sdk\samples\android-10 中的 NotePad 项目,打开其 AndroidManifest.xml 文件,代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) 2007 The Android Open Source Project

        Licensed under the Apache License, Version 2.0 (the "License");
        you may not use this file except in compliance with the License.
        You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

        Unless required by applicable law or agreed to in writing, software
        distributed under the License is distributed on an "AS IS" BASIS,
        WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
        See the License for the specific language governing permissions and
        limitations under the License.
-->
```

```
<!-- Declare the contents of this Android application.  The namespace
     attribute brings in the Android platform namespace, and the package
     supplies a unique name for the application.  When writing your
     own application, the package name must be changed from "com.example.*"
     to come from a domain that you own or have control over. -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.notepad">

    <application android:icon="@drawable/app_notes"
        android:label="@string/app_name">
        <provider android:name="NotePadProvider"
            android:authorities="com.example.notepad.provider.NotePad" />

        <activity android:name="NotesList"
            android:label="@string/title_notes_list">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <action android:name="android.intent.action.EDIT" />
                <action android:name="android.intent.action.PICK" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.GET_CONTENT" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
            </intent-filter>
        </activity>

        <activity android:name="NoteEditor"
            android:theme="@android:style/Theme.Light"
            android:configChanges="keyboardHidden|orientation">
            <!-- This filter says that we can view or edit the data of
                 a single note -->
            <intent-filter android:label="@string/resolve_edit">
                <action android:name="android.intent.action.VIEW" />
                <action android:name="android.intent.action.EDIT" />
                <action android:name="com.android.notes.action.EDIT_NOTE" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
            </intent-filter>

            <!-- This filter says that we can create a new note inside
                 of a directory of notes. -->
            <intent-filter>
                <action android:name="android.intent.action.INSERT" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
            </intent-filter>

        </activity>
```

```
72
73          <activity android:name="TitleEditor"
74              android:label="@string/title_edit_title"
75              android:theme="@android:style/Theme.Dialog"
76              android:icon="@drawable/ic_menu_edit"
77              android:windowSoftInputMode="stateVisible">
78              <!-- This activity implements an alternative action that can be
79                   performed on notes: editing their title.  It can be used as
80                   a default operation if the user invokes this action, and is
81                   available as an alternative action for any note data. -->
82              <intent-filter android:label="@string/resolve_title">
83                  <!-- This is the action we perform.  It is a custom action we
84                       define for our application, not a generic VIEW or EDIT
85                       action since we are not a general note viewer/editor. -->
86                  <action android:name="com.android.notepad.action.EDIT_TITLE" />
87                  <!-- DEFAULT: execute if being directly invoked. -->
88                  <category android:name="android.intent.category.DEFAULT" />
89                  <!-- ALTERNATIVE: show as an alternative action when the user is
90                       working with this type of data. -->
91                  <category android:name="android.intent.category.ALTERNATIVE" />
92                  <!-- SELECTED_ALTERNATIVE: show as an alternative action the user
93                       can perform when selecting this type of data. -->
94                  <category android:name="android.intent.category.SELECTED_ALTERNATIVE" />
95                  <!-- This is the data type we operate on. -->
96                  <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
97              </intent-filter>
98          </activity>
99
100         <activity android:name="NotesLiveFolder"
101             android:label="@string/live_folder_name"
102             android:icon="@drawable/live_folder_notes">
103             <intent-filter>
104                 <action android:name="android.intent.action.CREATE_LIVE_FOLDER" />
105                 <category android:name="android.intent.category.DEFAULT" />
106             </intent-filter>
107         </activity>
108
109     </application>
110
111     <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="4"/>
112 </manifest>
```

(1) 第 2～21 行是关于这个示例的注释，说明这个示例的权属和版本，以及出处等信息。

(2) 第 22～112 行定义<manifest>节点。

(3) 第 25～109 行定义<application>节点，在其中定义了一个<provider>节点，4 个<activity>节点，这说明至少有 5 个 Java 代码文件 NotePadProvider. java、NotesList. java、NoteEditor. java、TitleEditor. java 和 NotesLiveFolder. java。

(4) 第 27～28 行声明了一个 ContentProvider 组件。

(5) 第 30～48 行声明了一个 Activity 组件，其中设置了三个 IntentFilter 过滤信息。

(6) 第 50～71 行声明了一个 Activity 组件，其中设置了两个 IntentFilter 过滤信息。

(7) 第 73～98 行声明了一个 Activity 组件，其中设置了一个 IntentFilter 过滤信息。

(8) 第 100～107 行声明了一个 Activity 组件，其中设置了一个 IntentFilter 过滤信息。

(9) 第 111 行声明 Android SDK 的最低版本号和与目标设备的 API 版本相匹配的版本号，指定 minSdkVersion 主要是为了声明该应用程序的兼容性。

小结

本章主要介绍了 Android 应用程序的基本架构、基本组件，结合 HelloAndroid 项目比较详细地解析了 Android 应用程序的组成。由于功能清单文件 AndroidManifest. xml 在 Android 应用项目中的至关重要性，专门拿出来进行详细说明。相信读者已经对 Android 应用程序的组成有了更进一步的认识。

当然，想要知道 Android 系统是如何对应用程序的这些组件、控件及数据资源进行控制和信息通信的？将在第 3 章中进行介绍。

练习

理解 Android SDK 中例程的目录结构和 AndroidManifest. xml 文件。例程路径：SDK 文件夹下的 samples 子文件夹中，本书是 E:\android-sdk\samples。

第3章 Android应用程序的控制机制

Android 应用程序由 Activity、Service、Broadcast Receiver 和 Content Provider 等组件组成，其中 Activity 是使用频率最高、最重要的组件。在 Android 应用中，Activity 提供可视化的用户界面，一个 Android 应用通常由多个 Activity 组成。每个 Activity 有自己的生命周期，一个 Activity 组件结束，另一个 Activity 将处于活动状态，它们组成了一个应用任务。每个 Activity 的状态由 Android 系统来控制。想要知道 Android 应用程序是如何运行控制的，首先必须明白 Activity 组件在应用程序中的控制机制。

3.1 Android 应用程序的界面

Activity 是 Android 应用程序与使用者互动的主要元素，当使用者开启一个应用程序时，第一个看到的画面就是一个 Activity。在 Android 中，每一个 Activity 就是一个单独的屏幕显示。每个 Activity 组件由 XML 布局文件、Java 代码文件以及 Activity 组件状态来确定它在屏幕上显示的内容、操作响应和显示的时机。当然，别忘了一定要在 AndroidManifest. xml 中声明它，否则将看不到它。

通常把 Activity 中的内容在屏幕上的显示称作用户界面(User Interface，UI)。在用户界面中可显示的内容有很多，如文本框、按钮、列表框、图片、进度条等，这些用户界面元素称为控件。大部分控件是可见的。在 Android 中，所有的可视控件都继承自 View 类。View 类提供了绘制和事件处理的方法。

对于 Activity 中控件的绘制，既可以使用 XML 文件描述，也可以通过成员方法在 Java 代码中动态设置。但本着 MVC 的设计思想，通常是使用 XML 文件来描述界面的布局，而对于 Activity 组件中的屏幕控件对象的处理与控制，则使用 Java 代码进行 Activity 类的定义。

使用 XML 声明法来描述应用程序的可视控件及其布局信息，此文件称为布局文件。每一个 Activity 对应一个布局文件，所有布局文件都存放在应用项目目录下的“res/layout”子目录内。布局文件只定义了 Activity 的显示内容，但并没有告诉 Android 什么时候显示它。

在应用项目目录下的“src”目录的包内，存放定义此 Activity 类的 Java 代码文件。在代码文件中定义 Activity 的显示时机，以及显示、退出时 Activity 状态信息的保存与恢复，用户交互操作时各控件的事件响应等控制逻辑。每个定义的 Activity 类都继承自 android. app. Activity，所以在其他类的定义代码文件前部都必须加上“import android. app. Activity;”。

一个 Activity 可以启动另外一个，甚至包括与它不在同一应用程序之中的 Activity。举个例子，假设想让用户看到某个地方的街道地图，而已经存在一个具有此功能的 Activity 了，那么你的 Activity 所需要做的工作就是把请求信息放到一个 Intent 对象里面，并把它传递给 startActivity()，于是地图浏览器就会显示那个地图。而当用户按下 BACK 键的时候，你的

Activity 又会再一次显示在屏幕上。对于用户来说，这看起来就像是地图浏览器是你 Activity 所在的应用程序中的一个组成部分，其实它是在另外一个应用程序中定义，并运行在那个应用程序的进程之中的。

关于 Android 中的组件和应用，之前涉及的大都是静态的概念。而当一个应用运行起来，就难免会需要关心进程、线程这样的概念。

3.2　Android 应用程序的任务、进程和线程

在 Android 中，组件的动态运行有一个最与众不同的概念，就是任务(Task)。任务的介入，最主要的作用是将组件之间的连接从进程概念的细节中剥离出来，可以以一种不同模型的东西进行配置，在很多时候，能够简化上层开发人员的理解难度，帮助读者更好地进行开发和配置。

3.2.1　任务

完成用户的一个目的的所有 Activity 组成一个任务。确切地说，任务是一组以栈的模式将这些 Activity 组件聚集在一起的集合，这个栈称作任务栈(Task Stack)。Android 系统用一个任务栈来记录一个任务，它们有潜在的前后驱关联，新加入的 Activity 组件，位于栈顶，并仅有在栈顶的 Activity，才会有机会与用户进行交互。而当栈顶的 Activity 完成使命退出的时候，任务会将其退栈，并让下一个将跑到栈顶的 Activity 来与用户面对面，直至栈中再无更多的 Activity，任务结束。

以收发邮件任务为例，通常接收并回复邮件会依次进行下述操作，描述如下。

(1) 首先是单击 E-mail 应用，进入收件箱，这部分由 Activity A 完成。

(2) 选中一封邮件，单击查看详情，这部分由 Activity B 完成。

(3) 单击“回复”，开始写新邮件，这部分由 Activity C 完成。

(4) 写几行字，选择联系人，进入选择联系人界面，这部分由 Activity D 完成。

(5) 选择好联系人，继续写邮件。

(6) 写好邮件，发送，回到原始邮件。

(7) 返回到收件箱。

(8) 退出 E-mail 程序。

那么，任务栈会随着操作事件的变化而发生相应的变化，其过程如表 3-1 所示。

表 3-1　收发邮件任务栈的动态变化

事　件	Task 栈
单击 E-mail 应用，进入收件箱	A
选中一封邮件，单击查看详情	A B
单击“回复”，开始写新邮件	A B C
写几行字，选择联系人，进入选择联系人界面	A B C D
选择好联系人，继续写邮件	A B C
写好邮件，发送，回到原始邮件	A B
返回到收件箱	A
退出 E-mail 程序	null

表 3-1 描述了收发邮件的一个实例。从用户一进入邮箱开始，到回复完成，退出应用整个过程的任务栈变化。这是一个标准的栈模式，对于大部分的状况，这样的任务模型足以应付，但是在实际开发中，还要考虑到应用的性能、开销等问题。例如，启动一个浏览器，在 Android 中是一个比较沉重的过程，它需要做很多初始化的工作，并且会有不小的内存开销。但与此同时，用浏览器打开一些内容，又是一般应用都会有的一个需求。试设想一下，如果同时有 10 个运行着的应用都需要启动浏览器，就会有 10 个任务栈都堆积着很雷同的浏览器 Activity，这将是一个多么奢侈的浪费！于是，可以这样设想，浏览器 Activity 可不可以作为一个单独的任务而存在，不管是来自哪个任务的请求，浏览器的任务都不会归并过去。这样，虽然浏览器 Activity 本身需要维系的状态更多了，但整体的开销将大大减少。

3.2.2 进程

进程是低级核心处理过程，用于运行应用程序代码。当某个组件第一次运行的时候，Android 就启动了一个进程。默认情况下，所有的组件和程序运行在这个进程和线程中。

当然，也可以安排组件在其他的进程或者线程中运行进程，组件运行的进程由 Androidmanifest 文件控制。组件的标签<activity>，<service>，<receiver>和<provider>都包含一个 process 属性。这个属性可以设置组件运行的进程：可以配置组件在一个独立进程运行，或者多个组件在同一个进程运行。甚至可以多个程序在一个进程中运行，前提是如果这些程序共享一个 UserID，并且给定同样的权限。< application>标签也包含 process 属性，用来设置程序中所有组件的默认进程。

所有的组件在此进程的主线程中实例化，系统对这些组件的调用从主线程中分离。并非每个对象都会从主线程中分离。一般来说，响应例如 View. onkeydown()用户操作的方法和通知的方法也在主线程中运行。这就表示，组件被系统调用的时候不应该长时间运行或者阻塞操作(如网络操作或者计算大量数据)，因为这样会阻塞进程中的其他组件。可以把这类操作从主线程中分离。

当更加常用的进程无法获取足够内存，Android 可能会关闭不常用的进程。下次启动程序的时候会重新启动进程。在决定哪个进程需要被关闭的时候，Android 会考虑哪个对用户更加有用。如 Android 会倾向于关闭一个长期不显示在界面的进程来支持一个经常显示在界面的进程。是否关闭一个进程决定于组件在进程中的状态。

1. 进程分类

Android 系统会根据运行于进程内的组件和组件的状态把进程置于 5 种不同的重要性等级。当需要系统资源时，重要性等级越低的先被淘汰。下面按重要性等级由高到低介绍这 5 类进程。

1) 前台进程

用户当前正在做的事情需要这个进程。如果满足下面的条件，进程即为前台进程。

(1) 这个进程拥有一个正在与用户交互的 Activity(这个 Activity 的 onResume()方法被调用)。

(2) 这个进程拥有一个绑定到正在与用户交互的 Activity 上的 Service。

(3) 这个进程拥有一个前台运行的 Service，并且这个 Service 调用了方法 startForeground()。

(4) 这个进程拥有一个正在执行其任何一个生命周期回调方法(onCreate()，onStart()或

onDestroy())的 Service。

(5) 这个进程拥有正在执行其 onReceive()方法的 BroadcastReceiver。

通常,在任何时间点,只有很少的前台进程存在。一般情况下,前台进程不会被“杀死”,它们只有在达到无法调和的矛盾时,例如内存太小而不能继续运行时,才会被杀。通常,到了这时,设备就达到了一个内存分页调度状态,所以需要杀一些前台进程来保证用户界面的正常反应。

2) 可见进程

一个进程不拥有运行于前台的组件,但是依然能被用户所见。满足下列条件时,进程即为可见进程。

(1) 这个进程拥有一个不在前台但仍可见的 Activity(它的 onPause()方法被调用)。例如,当一个前台 Activity 启动一个对话框时,就出现这种情况。

(2) 一个可视的 Activity 所绑定的 Service。

可见进程被认为是极其重要的。并且,除非只有杀掉它才可以保证所有前台进程的运行,否则是不能动它的。

3) 服务进程

满足下列条件时,进程即为服务进程。

(1) 一个由 startService()方法启动的 Service。

(2) 支持正在处理的不需要可见界面运行的 Service。

尽管一个服务进程不直接被用户所见,但是它们通常做一些用户关心的事情(例如播放音乐或下载数据),所以系统不到前台进程和可见进程活不下去时不会杀它。

4) 后台进程

满足下列条件时,进程即为服务进程。

(1) 一个进程拥有一个当前不可见的 Activity(Activity 的 onStop()方法被调用)。

(2) 目前没有服务的 Service。

这样的进程不会直接影响到用户体验,所以系统可以在任意时刻杀掉它们从而为前台、可见以及服务进程们提供存储空间。通常有很多后台进程在运行。它们被保存在一个 LRU(最近最少使用)列表中,Android 将会使用 last-seen-first-killed 模式“杀死”进程来为前台进程、可见进程和服务进程获得资源。如果一个 Activity 正确地实现了它的生命周期方法,并保存了它的当前状态,那么杀死它的进程将不会对用户的可视化体验造成影响。因为当用户返回到这个 Activity 时,这个 Activity 会恢复它所有的可见状态。

5) 空进程

空进程不拥有任何 Active 组件,对用户没有任何作用。

为了改善系统的整体性能,Android 通常在内存中保留生命周期结束了的应用。保留这类进程的唯一理由是高速缓存,这样可以提高下一次一个组件要运行它时的启动速度。通常根据进程高速缓存和底层的内核高速缓存之间的整体系统资源的需要而杀死它们。

2. 进程作用

在 Android 操作系统中,进程完全是应用程序的具体实现,不是用户通常认为的那种东西。它们的主要用途是简单的。

(1) 改善稳定性或者安全性,通过把未信任或者不稳定的代码放入独立的进程的方法来

解决。

(2) 简化在同一进程中多个.apk 文件的代码的运行。

(3) 有助于系统管理资源,通过把重量级代码放入独立的进程,可以被杀掉,而且和程序的其他部分无关。

问一下:

什么是.apk 文件?

APK 是 AndroidPackage 的缩写,即 Android 安装包,它也是发布应用程序的文件,当用户在其设备上安装应用程序时,由用户下载这个文件。APK 类似 Windows 的 exe 文件,每个要安装到 Android 平台的应用都要被编译打包为一个单独的文件,后缀名为.apk。通过将 APK 文件直接传到 Android 模拟器或 Android 手机中执行即可安装。

APK 文件是由 Eclipse 编译生成的文件包,其中包含应用程序的二进制代码、资源、配置文件等。APK 文件其实是 zip 格式,但其后缀名被修改为 apk。通过 WinRAR 或 UnZip 解压后,可以看到 Dex 文件。Dex 是 Dalvik VM executes 的简称,即 Android Dalvik 执行程序; Resources.arsc 文件,即编译后的二进制资源文件; AndroidManifest.xml 配置清单文件; META-INF 目录,该目录下存放的是签名信息; 等等。

3.2.3 线程

每个进程有一到多个线程运行在其中。在多数情况下,系统不会为进程中的每一个组件启动一个新的线程,进程中的所有组件都在 UI 线程中实例化,保证应用程序是单线程的,除非应用程序自己又创建了线程。例如,用户界面需要很快对用户进行响应,因此某些费时的操作,例如网络连接、下载或者非常占用服务器时间的操作应该放到其他线程中。

线程通过 Java 的标准对象 Thread 创建。Android 提供了很多方便的管理线程的方法,如 Looper 在线程中运行一个消息循环; Handler 传递一个消息; HandlerThread 创建一个带有消息循环的线程,等等。

要记住:

(1) 不要阻塞 UI 线程。如果在 UI 线程中执行阻塞或者耗时操作会导致 UI 线程无法响应用户请求。

(2) 不能在非 UI 线程(也称为工作线程)中更新 UI。这是因为 Android 的 UI 控件都是线程不安全的。

由上所述,开发者经常会启动工作线程完成耗时操作或阻塞操作,如果需要在工作线程的执行期间更新 UI 状态,则应该通知 UI 线程来进行。

编写 Android 平台的基本应用程序和编写桌面应用程序的难度,两者并没什么不同。甚至因为 Android 平台拥有开放、免费、跨平台的开发工具,使得 Android 平台应用程序的开发更为单纯。

但是请别忘了,Android 平台也是个手机操作系统。撇掉其他功能不谈,手机的特性就是应该能够随时在未完成目前动作的时候,离开正在使用的功能,切换到接电话、接收简讯模式,而且在接完电话回来应用程序时,还希望能看到离开时的内容。这就是所谓的多任务(Multi-Task)的操作系统。同时执行多个程序有它的明显好处,但是也有它的严重缺点。每多执行一个应用程序,就会多耗费一些系统内存,而手机里的内存是相当有限的。当同时执行的程序过多,或是关闭的程序没有正确释放掉内存,执行系统时就会觉得越来越慢,甚至不稳定。为

了解决这个问题，Android 引入了一个新的机制——生命周期(Life Cycle)。

3.3 Android 应用程序生命周期

应用程序进程从创建到结束的全过程就是应用程序的生命周期。与其他系统不同的是，Android 应用程序的生命周期是由 Android 框架进行管理，而不是由应用程序直接控制。一般情况下，Android 系统会根据应用程序对用户的重要性和当前系统的负载来决定生命周期的长短。

对于应用开发者来说，理解不同的应用组件(特别是 Activity、Service、Intent 和 Receiver)对应用进程的生命周期的影响，这是非常重要的。如果没有正确地使用这些组件，将会导致当应用正在处理重要的工作时，进程却被系统销毁的后果。

应用程序组件有其生命周期：由 Android 初始化它们，以响应 Intent 响应意图，直到结束，实例被销毁。在这期间，它们有时候处于激活状态，有时候处于非激活状态；对于 Activity(活动)而言，用户有时候可见，有时候不可见。Activity 类是 Android 应用生命周期的重要部分之一，本节主要讨论 Activity 的生命周期及它们可能的状态。

3.3.1 Activity 的生命周期

在系统中的 Activity 被一个 Activity 栈所管理。当一个新的 Activity 启动时，将被放置到栈顶，成为运行中的 Activity，前一个 Activity 保留在栈中，不再放到前台，直到新的 Activity 退出为止。从启动到退出，Activity 经历了一个生命周期。

1. Activity 生命周期的状态

Activity 生命周期中存在 5 种状态：启动，运行，暂停，停止，销毁。

当 Activity 被压入栈顶，在屏幕的前台，称为启动状态(Starting)。此时，Activity 可见并获得焦点，与用户进行交互，称为运行状态(Running)。一般地，当一个 Activity 启动后随即处于运行状态。

如果一个 Activity 失去焦点，但是依然可见，例如一个新的非全屏的 Activity 或者一个透明的 Activity 被放置在栈顶时，称为暂停状态(Paused)。一个暂停状态的 Activity 依然保持活力，如保持所有的状态、成员信息和窗口管理器保持连接等。但是在系统内存极端低下的时候将被杀掉。

如果一个 Activity 被另外的 Activity 完全覆盖掉，称为停止状态(Stopped)。它依然保持所有状态和成员信息，但是它不再可见，所以它的窗口被隐藏。当系统内存需要被用在其他地方的时候，Stopped 的 Activity 将被杀掉。

如果一个 Activity 是 Paused 或者 Stopped 状态，系统可以将该 Activity 从内存中删除，称为销毁状态(Destroyed)。Android 系统采用两种方式进行删除，要么要求该 Activity 结束，要么直接杀掉它的进程。当该 Activity 再次显示给用户时，它必须重新开始和重置前面的状态。

2. Activity 生命周期的状态转变

Activity 在其生命周期中，上述几种状态可以通过操作事件或内部控制机制进行相互转

变，具体的状态转变以及触发事件如图 3-1 所示。

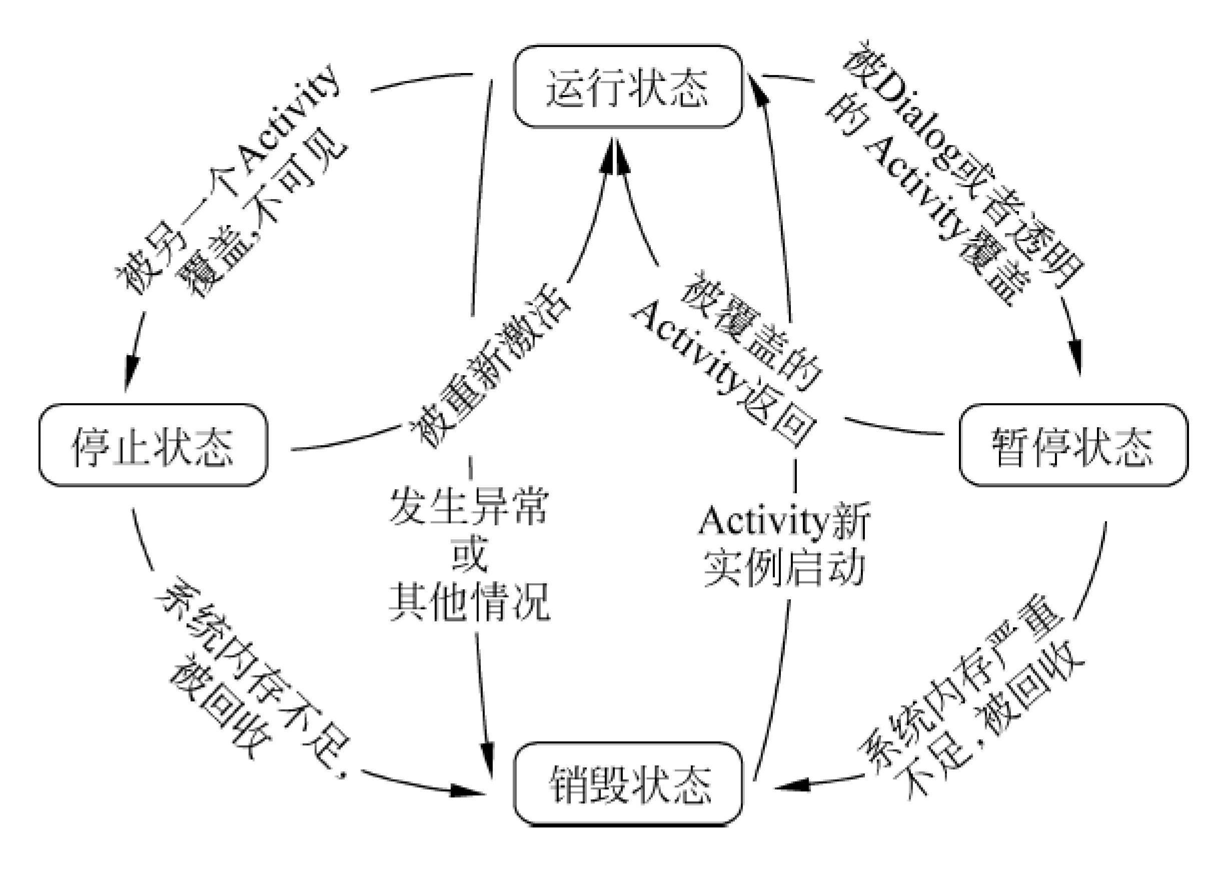


图 3-1 Activity 的状态转变

3.3.2 Activity 生命周期中的方法

在前面，我们发现所有继承自 Activity 的类都重写了 onCreate()方法，程序运行就会自动进入这个方法。其实 Activity 类中还有很多类似于 onCreate()的方法，例如 onStart()、onResume()、onPause()、onDestroy()等，而这些方法都是系统自动调用。

当一个 Activity 从一个状态转变为另一个状态时，Activity 被这些方法所通知。我们可以重写所有这些方法以在 Activity 状态改变时进行合适的工作。所有的 Activity 都必须实现 onCreate() 用以当对象第一次实例化时进行初始化设置。在实际开发中，很多时候 Activity 需要重写 onPause()，以提交数据变化或准备停止与用户的交互。

在 Activity 的整个生命周期中有 9 种方法。表 3-2 对每个方法进行了详细的描述以及在 Activity 的整个生命周期中的定位。

表 3-2 生命周期方法说明

方　法	描　述	可被杀死	下一个
onCreate()	在 Activity 第一次被创建的时候调用。这里是做所有初始化设置的地方，如创建视图、绑定数据至列表等。如果曾经有状态记录，则调用此方法时会传入一个包含着此 Activity 以前状态的 Bundle 对象作为参数。 紧跟其后的方法总是 onStart()	否	onStart()
onRestart()	在 Activity 停止后，在再次启动之前被调用。 紧跟其后的方法总是 onStart()	否	onStart()
onStart()	当 Activity 变为用户可见之前被调用。 当 Activity 转向前台时接下来调用 onResume()，在 Activity 变为隐藏时接下来调用 onStop()	否	onResume()或 onStop()
onResume()	在 Activity 开始与用户进行交互之前被调用。此时 Activity 位于堆栈顶部，并接受用户输入。 紧跟其后的方法是 onPause()	否	onPause()

续表

方　法	描　述	可被杀死	下一个
onPause()	当系统将要启动另一个 Activity 时调用。此方法主要用来将未保存的变化进行持久化，停止动画和其他耗费 CPU 的动作等。这一切动作应该在短时间内完成，因为下一个 Activity 必须等到此方法返回后才会继续。 当 Activity 重新回到前台时接下来调用 onResume()。当 Activity 变为用户不可见时接下来调用 onStop()	是	onResume()或 onStop()
onStop()	当 Activity 不再为用户可见时调用此方法。这可能发生在它被销毁或者另一个 Activity(可能是现存的或者是新的)回到运行状态并覆盖了它。 如果 Activity 再次回到前台与用户交互则接下来调用 onRestart()，如果关闭 Activity 则接下来调用 onDestroy()	是	onRestart()或 onDestroy()
onDestroy()	在 Activity 销毁前调用。这是 Activity 接收的最后一个调用。这可能发生在 Activity 结束(调用了它的 finish() 方法)或者因为系统需要空间所以临时销毁了此 Acitivity 的实例时。可以通过 isFinishing() 方法来区分这两种情况	是	nothing
onSaveInstanceState (Bundle)	调用该方法让 Activity 可以保存每个实例的状态。 当 Activity 由运行状态变为可见时接下来调用 onPause()，当 Activity 由暂停状态变为可见停止状态时接下来调用 onStop()		onPause() 或 onStop()
onRestoreInstanceState (Bundle)	使用 onSaveInstanceState()方法保存的状态来重新初始化某个 Activity 时调用该方法。 紧跟其后的方法是 onResume()		onResume()

这 9 个方法定义了 Activity 的整个生命周期。有三个嵌套的循环，用户可以通过这 9 个方法监视以下内容。

(1) Activity 的整个生命时间。

从第一次调用 onCreate()开始直到调用 onDestroy()结束。一个 Activity 在 onCreate()中做所有的“全局”状态的初始设置，在 onDestroy()中释放所有保留的资源。举例来说，有一个线程运行在后台从网络上下载数据，它可能会在 onCreate()中创建线程，在 onDestroy()中结束线程。

(2) Activity 的可视生命时间。

从调用 onStart() 到相应的调用 onStop()。在这期间，用户可以在屏幕上看见 Activity，虽然它可能不是运行在前台且与用户交互。在这两个方法之间，可以保持显示 Activity 所需要的资源。举例来说，可以在 onStart()中注册一个广播接收者监视影响你的 UI 的改变，在 onStop()中注销。因为 Activity 在可视和隐藏之间来回切换，onStart()和 onStop()可以调用多次。

(3) Activity 的前台生命时间。

从调用 onResume() 到相应的调用 onPause()。在这期间，频繁地在重用和暂停状态转换。例如，当设备进入睡眠状态或一个新的 Activity 启动时调用 onPause()，当一个 Activity

返回或一个新的 Intent 被传输时调用 onResume()。因此,这两个方法的代码应当是相当轻量级的。

图 3-2 解释了这三个循环和状态之间状态的可能路径,其中标注了“ * ”号的为可选项,从一个状态到另一个状态之间,用(1)、(2)、(3)表示调用方法的先后顺序。

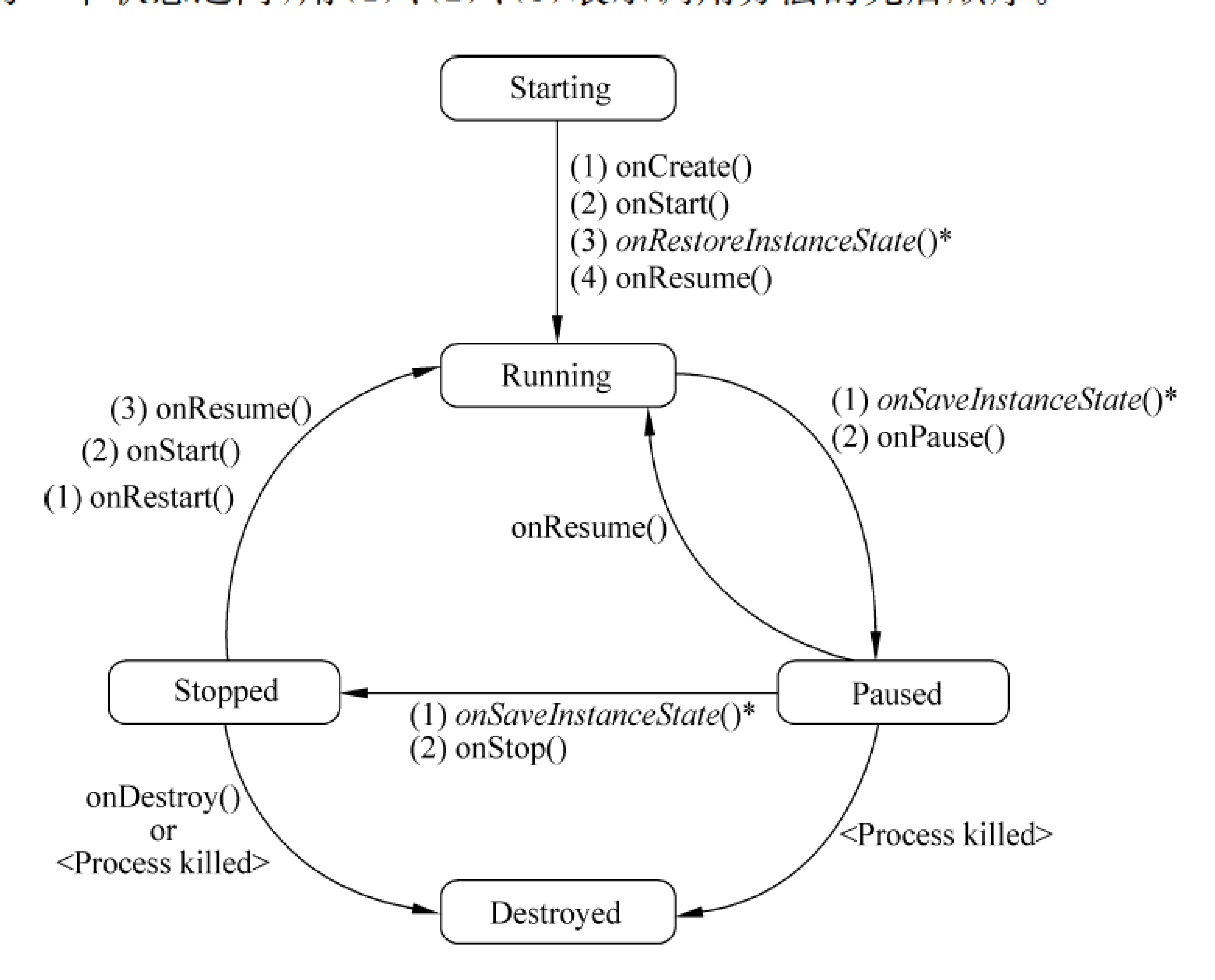


图 3-2 在生命周期中成员方法的调用时机

多个 Activity 生命周期中方法的调用过程举例。如果有两个 Activity 分别为 Activity A、Activity B,依次进行下列操作,Android 的生命周期方法调用的情况如下。

(1) 先启动第一个界面 Activity A,方法回调的次序是:

onCreate(A)→onStart(A)→onResume(A)

(2) 如果 Activity A 不关闭,跳转第二个 Activity B,方法回调的次序是:

onFreeze(A)→onPause(A)→onCreate(B)→ onStart(B)→onResume(B)→onStop(A)

(3) 如果按 Back 键后,回到第一个界面,这时方法回调的次序是:

onPause(B)→onActivityforResult(A)→onRestart(A)→onStart(A)→onResume(A)→onStop(B)→onDestroy(B)

(4) 如果单击 Exit 退出应用时,方法回调的次序是:

onPause(A)→ onStop(A)→onDestroy(A)

3.4 Android 组件间的通信

在 2.3 节中已知道,Intent 是连接应用程序的三个核心组件——Activity、Service 和 BroadcastReceiver 的桥梁,可想而知,Intent 把握着 Android 各组件间的通信。

刚接触 Intent 的时候,大多数人都会有点困惑:从字面上来说,它表示一种意图和目的;从使用上看,它似乎总是用于 Activity 之间的切换;而从它所在包 android.content 来看,它

似乎与内容有关。所以可以这样来理解它：Intent 类绑定一次操作，它负责携带这次操作所需要的数据以及操作的类型等。

Intent 本身是一个 Intent 类对象，Intent 类都定义在 android.content.Intent 中，所以在使用它之前需要在代码文件的前部加上“import android.content.Intent;”。每个 Intent 对象是由一个包含被执行操作抽象描述的被动的数据结构，或者对于广播而言，是某件已经发生并被声明的事情的描述。

Intent 存在不同的机制来传送 Intent 到每种组件中。

(1) 一个 Intent 对象通过调用 Context.startActivity()或者 Activity.startActivityForResult()来启动一个 Activity 或者让一个存在的 Activity 去做某些新的事情。

(2) 一个 Intent 对象通过调用 Context.startService()来发起一个 Service 或者递交新的指令给运行中的 Service。类似地，一个 Intent 对象通过 Context.bindService()来在调用组件和一个目标 Service 之间建立连接。作为一个可选项，如果一个 Service 还没运行它可以发起之。

(3) 一个 Intent 对象通过调用 Context.sendBroadcast()、Context.sendOrderedBroadcast()或者 Context.sendStickyBroadcast()传递给所有感兴趣的 BroadcastReceiver。许多种广播产生于系统代码。

这样，Android 系统会找到合适的 Activity、Service 或 BroadcastReceiver 来回应这个 Intent，必要时实例化它们。这些消息传送系统没有重叠：一个传送给 startActivity()的 Intent 是只会被传递给一个 Activity，永远不会给一个 Service 或广播接收者，广播意图仅被传递给 BroadcastReceiver，永远不会给 Activity 或者 Service，以此类推。

3.4.1 Intent 对象

一个 Intent 对象(Intent Objects)其实就是一堆信息的捆绑。它包含接收这个 Intent 的组件感兴趣的信息，例如将要采取的动作和操作的数据，再加上 Android 系统感兴趣的信息，例如应该处理这个 Intent 的组件类别和如何启动一个目标 Activity 的指令。Intent 对象由组件名称、Action、Data、Category、Extra 及 Flag 6 部分组成。下面分别给予描述。

1. 组件名称

组件名称(Component name)是要处理这个 Intent 的组件的名字。组件名字是可选的。如果被设置了，这个 Intent 对象将被传递到指定的类。如果没有设置，则在 Androidmanifest.xml 中，通过使用 IntentFilter 来找与该 Intent 最合适的组件。

组件名字通过方法 setComponent()、setClass()或者 setClassName()来设置，通过方法 getComponent()来读取。

2. 动作

动作(Action)是一个将被执行的动作的字符串命名，或者，对于广播意图而言，是发生并被报告的动作。这个 Intent 类定义了一些动作常量，用大写英文单词和下划线组成，如表 3-3 所示。

表 3-3 常用的动作常量说明

常 量	目标组件	动作含义
ACTION_CALL	Activity	拨打电话,被呼叫的联系人在数据中指定
ACTION_EDIT	Activity	显示数据给用户进行编辑
ACTION_GET_CONTENT	Activity	让用户选择数据并返回
ACTION_INSERT	Activity	在容器中插入一个空项
ACTION_MAIN	Activity	启动一个任务的起始 Activity,没有数据输入和数据返回
ACTION_PICK	Activity	从数据中选择一个子项目,并返回所选中的项目
ACTION_BATTERY_LOW	BroadcastReceiver	提示电池电量低
ACTION_SCREEN_ON	BroadcastReceiver	屏幕已开启
ACTION _HEADSET_PLUG	BroadcastReceiver	耳机插拔
ACTION _TIMEZONE_CHANGED	BroadcastReceiver	时区变化

动作很大程度上决定了 Intent 其他部分如何被组织,尤其是数据 data 和附加字段 extras。这就好比一个方法名决定了一些参数和返回值一样。因此,在实际开发中要尽可能地把具体的动作名与 Intent 的其他字段紧密联系起来。换句话说,要为组件能处理的 Intent 对象定义一个整体的协议,而不是定义一个孤立的动作。

开发者可以定义自己的动作,并定义相应的 Activity 来处理自定义动作。一个 Intent 对象里的动作可以通过 setAction()方法来设置,通过 getAction()方法来读取。

3. 数据

数据(Data)是为动作提供要操作的信息,用指向数据的一个资源标识符(URI)来表示。不同的动作伴随着不同种类的数据规格。例如,如果动作是 ACTION_EDIT,数据字段会包含可编辑文档的 URI;如果动作是 ACTION_CALL,数据字段会是一个含呼叫电话号码的 URI。当匹配一个 Intent 到一个能处理数据的组件时,除了它的 URI 外,通常需要知道数据类型(即多用途互联网邮件扩展,MIME)。

从 2.3 节已经知道,URI 的格式为“scheme://host:port/path”。在很多情况下,这个数据类型可以从 URI 里推断出来。尤其是当 scheme 的内容为“content”时,这意味着数据被存放在设备上而且由一个内容提供器控制着。

例如,指向 1 号联系人的 URI 为:content://contacts/1,在 Java 代码中,获取一个 URI 的语句格式为:

```
Uri uri = Uri.parse(<字符串>);
```

创建一个 Intent 对象的语句格式为:

```
Intent intent = new Intent(<动作>, <内容>);
```

代码:

```
Uri uri1 = Uri.parse("content://contacts/1");
Intent intent = new Intent(Intent. ACTION_VIEW, uri1);
```

“uri1”是一个 URI 变量,其值是“content://contacts/1”,它指向设备上(如手机)的联系

人信息集中的第一个联系人。新创建的 Intent 对象 intent 包含的信息是显示标识符为“1”的联系人的详细信息。

代码：

```
Uri uri = Uri.parse("content://contacts/");
Intent intent = new Intent(Intent. ACTION_PICK, uri);
```

“uri”是一个 URI 变量，其值是“content://contacts/”，它指向设备上（如手机）的所有联系人。新创建的 Intent 对象 intent 包含的信息是显示所有联系人的列表，并返回选择的联系人信息。

通过 setData()方法指定数据只能为一个 URI，通过 setType()方法指定数据只能是一个 MIME 类型，而通过 setDataAndType()方法可同时指定数据为 URI 和 MIME。通过 getData()方法来读取 URI，通过 getType()方法来读取类型。

4. 类别

类别（Category）是关于 Intent 中 action 要执行的动作的附加描述，它是一个字符串。可以把任意数目的类别描述放到一个 Intent 对象里。和动作一样，Intent 类别定义了若干类别常量，如表 3-4 所示。

表 3-4　常用的类别常量说明

常　量	含　义
CATEGORY_ALTERNATIVE	在某种数据类型的项目上可以替代默认执行的动作。例如，一个联系人的默认动作是浏览它，替代的可能是去编辑或删除它
CATEGORY_BROWSABLE	目标 Activity 可以被浏览器安全地唤起来显示被一个链接所引用的数据。例如，一张图片或一条 E-mail 消息
CATEGORY_ DEFAULT	设置这个类别来让组件成为 Intent 过滤器中定义的 data 的默认动作。这对使用显式 Intent 启动的 Activity 来说也是必要的
CATEGORY_HOME	这个 Activity 将显示桌面，也就是用户开机后看到的第一个屏幕或者按 HOME 键时看到的屏幕
CATEGORY_LAUNCHER	这个 Activity 可以是一个任务的初始 Activity，并被列在应用程序启动器的顶层
CATEGORY_PREFERENCE	目标 Activity 是一个选择面板

通过 addCategory()方法在一个 Intent 对象中添加一个类别，通过 removeCategory()方法删除之前添加的类别，通过 getCategories()方法可以获取当前对象的所有类别。

5. 附加信息

附加信息（Extra）是要递交给 Intent 处理组件的附加信息键-值对。就像一些动作伴随着特定的数据 URIS 类型一样，一些动作也要伴随着特定的附加信息。例如，一个 TIMEZONE_CHANGED_ACTIONIntent 有一个“时区”附加信息用来区别新的时区，而 HEADSET_PLUG_ACTION 有一个“状态”附加字段表明耳机有没有插着，以及一个“名字”附加信息来表示耳机的类型。如果想要创建一个 SHOW_COLOR 动作，颜色的值将被设置在一个附加的键-值对中。

Intent 对象有一系列的 put...()方法来插入各种不同的附加数据，同样，有一系列的 get...()

方法来读取数据。这些方法与 Bundle 对象的方法相似。事实上,附加信息就是可以被当作一个 Bundle 对象,通过使用 putExtras()和 getExtras()方法来插入和读取。

6. 标志

标志(Flag)是各种类型的标志。许多标志用来指示 Android 系统如何去加载一个 Activity(例如,哪个是这个 Activity 应该归属的任务)和启动后如何对待它(例如,它是否属于当前 Activity 列表),所有这些列表都在 Intent 类中定义了。

Intent 可以分成两大类,一类是显式意图(Explicit Intent),另一类是隐式意图(Implicit Intent)。

显式意图,即指定了组件名称,通过名字指明目标组件。一般地,组件名称通常不为其他应用程序的开发者所了解,显式意图常被用作应用程序的内部消息,例如一个 Activity 启动一个附属 Service 或姊妹 Activity。

隐式意图,即不指定组件名称。隐式意图经常用来激活其他应用程序的组件。

Android 递交一个显式的 Intent 给一个指定目标类的实例。Intent 对象中的组件名称唯一地确定哪个组件应该获取这个 Intent。隐式 Intent 则需要一个不同的策略。在没有指定目标组件的情况下,Android 系统必须找到最合适的组件来处理这个 Intent,这就需要用到意图过滤器(Intent filters)了。

3.4.2 Intent 过滤器

为了通知系统它们可以处理哪些 Intent,Activity、Service 和 BroadcastReceiver 可以有一个或多个意图过滤器。每个过滤器描述组件的一个能力,一系列组件想要接收的 Intent。它实际上按照一个期望的类型来进行 Intent 滤入,同时滤出不想要的 Intent。要提醒读者注意的是,意图过滤器只针对隐式意图起作用。对于一个显式意图,它总能够被递交给它的目标组件,而无论它包含什么。这种情况下过滤器不起作用。

在 Android 系统中,可以用 Java 代码来设置 IntentFilter 类的一个实例,但是在更多情况下,是使用在应用程序清单文件 AndroidManifest.xml 中设置<intent-filter>标签。在其中设置需要过滤的内容。<intent-filter>标签常包含<action>、<category>、<data>等元素。

3.4.3 Intent 解析

通过比较 Intent 对象的内容和意图过滤器,找到相匹配的目标组件。如果一个组件没有任何的意图过滤器,那它只能接收显式意图。一个带过滤器的组件可以同时接收显式和隐式意图。

当一个 Intent 对象被一个意图过滤器测试时,一般是通过对动作、数据(URI 和 MIME)和类别三个方面进行监测的。下面分别进行介绍。

1. 检查 Action

一个 Intent 只能设置一种 Action,但是一个 IntentFilter 却可以设置多个 Action 过滤。当 IntentFilter 设置了多个 Action 过滤时,只需一个满足即可完成 Action 验证。如果

IntentFilter 中没有声明任何一个 Action，那么任何的 Action 都不能与之匹配。如果 Intent 中没有包含任何 Action，那么只要 IntentFilter 中含有 Action 时，便会与之匹配成功。

2. 检查 Data

对数据的监测主要包含两部分，一是对数据的 URI 进行监测，二是对数据的类型进行监测。而数据的 URI 又被分为三部分：scheme、authority、path，只有这些信息完全匹配时，Data 的验证才会成功。

3. 检查 Category

在 IntentFilter 中同样可以设置多个 Category。当 Intent 中的 Category 与 IntentFilter 中的一个 Category 完全匹配时，此 Category 便会验证通过，而其他的 Category 并不受影响。但是当 IntentFilter 中没有设置 Category 时，只能与没有设置 Category 的 Intent 匹配。

一个隐式意图为了递交到拥有这个过滤器的组件，它必须通过以上三项测试。即使只有一个不通过，Android 系统都不会把它递交给这个组件。应该注意的是，由于一个组件可以包含多个意图过滤器，一个 Intent 不能通过其中一个过滤器，但可能在另外的过滤器上获得通过。

下面以 SDK 的 NotePad 范例为例进行介绍。在自己的计算机上找到 NotePad 范例的位置，如本书的在 E:\android-sdk\samples\android-10\NotePad 中，打开 AndroidManifest.xml 文件，阅读第一个<activity>标签，它声明了三个过滤器，代码如下。

```
1  <activity android:name="NotesList"
2      android:label="@string/title_notes_list">
3      <intent-filter>
4          <action android:name="android.intent.action.MAIN" />
5          <category android:name="android.intent.category.LAUNCHER" />
6      </intent-filter>
7      <intent-filter>
8          <action android:name="android.intent.action.VIEW" />
9          <action android:name="android.intent.action.EDIT" />
10         <action android:name="android.intent.action.PICK" />
11         <category android:name="android.intent.category.DEFAULT" />
12         <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
13     </intent-filter>
14     <intent-filter>
15         <action android:name="android.intent.action.GET_CONTENT" />
16         <category android:name="android.intent.category.DEFAULT" />
17         <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
18     </intent-filter>
19 </activity>
```

(1) 第 3～6 行定义第一个过滤器，用于声明名称为“NotesList”的 Activity 是本应用程序的起始 Activity，并且是被并列在应用程序启动器的顶层。其中，元素“<action android:name="android.intent.action.MAIN" />”匹配 Intent 的动作常量“ACTION_MAIN”；元素“<category android:name="android.intent.category.LAUNCHER" />”匹配 Intent 的类别常量“CATEGORY_LAUNCHER”。

(2) 第 7～13 行定义第二个过滤器，用于声明用户可以查看、编辑或选择的一个指定的记

事目录。其中，元素“＜action android:name="android.intent.action.VIEW" /＞”匹配 Intent 的动作常量“ACTION_VIEW”，元素“＜action android:name="android.intent.action.EDIT" /＞”匹配 Intent 的动作常量“ACTION_EDIT”，元素“＜action android:name="android.intent.action.PICK" /＞”匹配 Intent 的动作常量“ACTION_PICK”；元素“＜category android:name="android.intent.category.DEFAULT" /＞”匹配 Intent 的类别常量“CATEGORY_DEFAULT”；元素“＜data android:mimeType="vnd.android.cursor.dir/vnd.google.note" /＞”匹配 Intent 的数据的 MIME 类型是“vnd.android.cursor.dir/vnd.google.note”。

(3) 在 NotesList.java 代码中，例如有如下代码创建一个 Intent 对象，其中 noteUri 是存放 URI 的一个变量。

```
startActivity(new Intent(Intent.ACTION_EDIT, noteUri));
```

(4) 第 14～18 行定义第三个过滤器，用于声明在没有初始目录说明的情况下查找一个特定的记录。其中，元素“＜action android:name="android.intent.action.GET_CONTENT" /＞”匹配 Intent 的动作常量“ACTION_GET_CONTENT”。

在第二个＜activity＞标签中，它声明了三个过滤器，代码如下。

```
1  <activity android:name = "NoteEditor"
2      android:theme = "@android:style/Theme.Light"
3      android:configChanges = "keyboardHidden|orientation">
4      <! -- This filter says that we can view or edit the data of
5           a single note -->
6      <intent-filter android:label = "@string/resolve_edit">
7          <action android:name = "android.intent.action.VIEW" />
8          <action android:name = "android.intent.action.EDIT" />
9          <action android:name = "com.android.notes.action.EDIT_NOTE" />
10         <category android:name = "android.intent.category.DEFAULT" />
11         <data android:mimeType = "vnd.android.cursor.item/vnd.google.note" />
12     </intent-filter>
13
14     <! -- This filter says that we can create a new note inside
15          of a directory of notes. -->
16     <intent-filter>
17         <action android:name = "android.intent.action.INSERT" />
18         <category android:name = "android.intent.category.DEFAULT" />
19         <data android:mimeType = "vnd.android.cursor.dir/vnd.google.note" />
20     </intent-filter>
21 </activity>
```

(1) 第 6～12 行定义第一个过滤器，用于声明用户可以在一个便签页中浏览或编辑其中的数据。其中，元素“＜action android:name="com.android.notes.action.EDIT_NOTE" /＞”是用户自定义的动作。

(2) 在 NoteEditor.java 代码中，有如下代码创建一个自定义的动作，其中第 23 行语句中的 setAction()是设置动作的方法。

```
1  final String action = intent.getAction();
2  if (Intent.ACTION_EDIT.equals(action)) {
3          //Requested to edit: set that state, and the data being edited.
4          mState = STATE_EDIT;
```

```
5          mUri = intent.getData();
6    } else if (Intent.ACTION_INSERT.equals(action)) {
7          //Requested to insert: set that state, and create a new entry
8          //in the container.
9          mState = STATE_INSERT;
10         mUri = getContentResolver().insert(intent.getData(), null);
11
12         //If we were unable to create a new note, then just finish
13         //this activity.  A RESULT_CANCELED will be sent back to the
14         //original activity if they requested a result.
15         if (mUri == null) {
16             Log.e(TAG, "Failed to insert new note into " + getIntent().getData());
17             finish();
18             return;
19         }
20
21         //The new entry was created, so assume all will end well and
22         //set the result to be returned.
23         setResult(RESULT_OK, (new Intent()).setAction(mUri.toString()));
24
25     } else {
26         //Whoops, unknown action!  Bail.
27         Log.e(TAG, "Unknown action, exiting");
28         finish();
29         return;
30     }
```

第 16～20 行定义第二个过滤器，用于声明用户可以在便签目录中新建一页便签。

在第三个<activity>标签，它声明了三个过滤器，代码如下。

```
1  <activity android:name = "TitleEditor"
2      android:label = "@string/title_edit_title"
3      android:theme = "@android:style/Theme.Dialog"
4      android:icon = "@drawable/ic_menu_edit"
5      android:windowSoftInputMode = "stateVisible">
6      <! -- This activity implements an alternative action that can be
7           performed on notes: editing their title.  It can be used as
8           a default operation if the user invokes this action, and is
9           available as an alternative action for any note data.  -->
10     <intent - filter android:label = "@string/resolve_title">
11         <action android:name = "com.android.notepad.action.EDIT_TITLE" />
12         <category android:name = "android.intent.category.DEFAULT" />
13         <category android:name = "android.intent.category.ALTERNATIVE" />
14         <category android:name = "android.intent.category.SELECTED_ALTERNATIVE" />
15         <data android:mimeType = "vnd.android.cursor.item/vnd.google.note" />
16     </intent - filter>
17 </activity>
```

(1) 在这个 Activity 中，实现一个编辑转变，从编辑便签内容转变到编辑便签的标题。

(2) 第 12～14 行声明了三个< category >，只需要 Intent 中的一个 Category 与其中之一匹配，便可以验证通过。

3.4.4 Intent 使用案例

下面通过一个简单的案例来介绍两个 Activity 组件是怎样通过 Intent 进行通信的。

【案例 3.1】 有两个 Activity：ActivityMain 和 ActivitySub。ActivityMain 为首次进行的 Activity，其中有个按钮，单击该按钮可跳转到 ActivitySub 上，并且在标题栏显示跳转信息；ActivitySub 上也有一个按钮，单击该按钮可以返回到 ActivityMain 上，且在标题栏显示跳转返回信息。如此循环往复。

【说明】 由于有些知识是后面章节中的内容，这里需要提前用到，在此只作简单介绍。

- ✍在两个 Activity 组件中都用到了按钮控件，所以在布局文件中，需要声明按钮控件，声明按钮的标签为<Button>。
- ✍在 Java 代码文件中，需要对按钮控件设置监听，使用方法 setOnClickListener()，以随时捕捉按钮的单击信号，一旦监听到按钮被单击，则执行 onClick()事件方法定义的操作。
- ✍在这个案例中，是两个 Activity 相互调用，在创建 Intent 对象时最好用显式的。在两个 Activity 相调中有可能需要返回信息，所以可能需要使用 startActivityForResult()方法来发送 Intent 对象。
- ✍关于 startActivityForResult()方法的使用：通常情况下，如果是从 A 发送 Intent 到 B，又从 B 返回到 A，并且还需要传递信息的应用，在 A 的代码中使用 startActivityForResult()方法发送 Intent 到 B，并且还需要重写 onActivityResult()方法用于处理返回的数据；在 B 的代码中使用 setResut()方法准备好要回传的数据，并且需要使用 finish()的方法来将打包好的数据发回给 A，并且运行 A 中的 onActivityResult()部分的代码。

startActivityForResult()方法的格式为：

```
startActivityForResult(Intent intent, Int requestCode)
```

onActivityResult()方法的格式为：

```
onActivityResult(int requestCode, int resultCode, Intent intent)
```

setResut ()方法的格式为：

```
setResut(int resultCode, Intent intent)
```

【开发步骤及解析】

(1) 在 Eclipse 中创建一个名为 Activity_Intent 的 Android 项目。其应用程序名为 Activity Intent，包名为 cn.com.sgmsc.Activity_Intent，Activity 组件名为 ActivityMain。

(2) 编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
  <Button android:id = "@ + id/button1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "进入 ActivitySub" />

</LinearLayout>
```

① 第 7～10 行声明一个按钮控件。

② 第 7 行定义这个按钮的 id 变量名为“button1”，并添加该按钮资源到 R. java 文件中。以便在 Java 代码中提供调用。

③ 第 10 行定义这个按钮上显示的文本内容为“进入 ActivitySub”。

(3) 在 res/layout 目录下新建一个 XML 文件，命名为 subactivity. xml。注意，在 res 下所有目录里的文件名都必须由小写字母或数字命名。代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3    android:orientation = "vertical" android:layout_width = "fill_parent"
4    android:layout_height = "fill_parent">
5
6    < Button android:id = "@ + id/button2"
7          android:layout_width = "wrap_content"
8          android:layout_height = "wrap_content"
9          android:text = "返回 ActivityMain" />
10
11 </LinearLayout >
```

第 6～9 行声明一个按钮控件。定义按钮 id 变量名为“button2”，并添加该按钮资源到 R. java 文件中。定义按钮显示的文本内容为“返回 ActivityMain”。

(4) 开发案例的主要逻辑代码，首先打开 src/cn. com. sgmsc. Activity_Intent 包下的 ActivityMain. java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.Activity_Intent;          //声明代码所在包

import android.app.Activity;
import android.content.Intent;
…… //该处省略了引入部分的类，可以从指定网站中下载源代码查阅
import android.widget.Button;

public class ActivityMain extends Activity {
    /* 声明变量 */
    OnClickListener listener1 = null;
    Button button1;
    static final int REQUEST_CODE = 1;

    /* 当 Activity 第一次创建时调用此方法 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    listener1 = new OnClickListener() {        //创建监听对象并定义其 onClick 事件
      public void onClick(View v) {
        Intent intent1 = new Intent(ActivityMain.this, ActivitySub.class);
        intent1.putExtra("activitymain", "从'ActivityMain'进入");
        startActivityForResult(intent1, REQUEST_CODE);
      }
    };
    setContentView(R.layout.main);             //启动 main.xml 定义的布局
    button1 = (Button) findViewById(R.id.button1);
                                   //将资源中 id 号为 button1 的按钮控件赋予按钮变量
    button1.setOnClickListener(listener1);     //为按钮 button1 设置监听
    setTitle("首次进入'ActivityMain'页面!"); //设置标题栏显示内容
```

```
29    }
30
31      /* 当 Activity 第一次创建时调用此方法 */
32      @Override
33      protected void onActivityResult(int requestCode, int resultCode, Intent data) {
34      if (requestCode == REQUEST_CODE) {
35            if (resultCode == RESULT_CANCELED)
36                  setTitle("取消");
37            else if (resultCode == RESULT_OK) {
38              String temp = null;
39               Bundle extras = data.getExtras();
40                    if (extras != null) {
41                      temp = extras.getString("store");
42                    }
43              setTitle("现在是在 ActivityMain 里:" + temp);
44            }
45          }
46    }
47 }
```

① 第 4 行引入 Intent 所在类，因为要使用 Intent 对象，所以在代码前部要引入该类。

② 第 18～24 行创建一个监听 listener1，并在创建的同时定义一个 onClick 事件(见第 19～23 行)。在这个事件中定义了当监听到按钮被单击之后，将会处理的一系列操作。

③ 第 20 行创建了一个 Intent 对象，其对象名为"intent1"，其动作值为"ActivityMain.this"，其内容值为"ActivitySub.class"，即它指定了组件的名称，是一个显式的 Intent。

④ 第 21 行向 intent1 中添加一附加信息，此附加信息是一组键-值对的 Bundle 信息，其名为"activitymain"，其值为"从'ActivityMain'进入"。

⑤ 第 22 行使用 startActivityForResult()方法发送 intent1 对象，并同时发送一个请求码 REQUEST_CODE 给 ActivitySub。

⑥ 第 33～46 行重写 onActivityResult()方法。通过判断请求码值和返回码值，来确定是否正确地获得回传数据，如果是正确的便取出回传数据，将它显示在标题栏信息中。

(5) 在 src/cn.com.sgmsc.Activity_Intent 包下创建名为 ActivitySub.java 的文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.Activity_Intent;

import android.app.Activity;
import android.content.Intent;
…… //该处省略了引入部分的类，可以从指定网站中下载源代码查阅
import android.widget.Button;

public class ActivitySub extends Activity {
  OnClickListener listener1 = null;
  Button button1;
    /* 当 Activity 第一次创建时调用此方法 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.subactivity);
        listener1 = new OnClickListener() {
          public void onClick(View v) {
```

```
18                Bundle bundle = new Bundle();
19                bundle.putString("store", "自'ActivitySub'返回");
20                Intent intent2 = new Intent();
21                intent2.putExtras(bundle);
22                setResult(RESULT_OK, intent2);
23                finish();
24            }
25        };
26        button1 = (Button) findViewById(R.id.button2);
27        button1.setOnClickListener(listener1);
28        String data = null;
29        Bundle extras = getIntent().getExtras();
30            if (extras != null) {
31                data = extras.getString("activitymain");
32            }
33        setTitle("现在是在'ActivitySub'里: " + data);
34        }
35 }
```

① 第 16～25 行创建一个监听 listener1，并在创建的同时定义一个 onClick 事件(见第 17～24 行)。在这个事件中定义了，当监听到按钮被单击之后将会处理的一系列操作。注意，在 ActivitySub 第一次创建时不会运行其中的代码。

② 第 18 行和第 19 行创建一个 Bundle 对象，名为 bundle，并将一组键-值对保存到其中。其名为“store”，其值为“自'ActivitySub'返回”。

③ 第 20 行创建了一个 Intent 对象，其对象名为“intent2”，其内无任何信息。

④ 第 21 行向 intent2 中添加一附加信息，此附加信息是已保存 bundle 中的键-值对信息。

⑤ 第 22 行是打包好要回传的数据。在这里回传一个返回码值 RESULT_OK 和 intent2 对象。

⑥ 第 23 行将打包好的数据发回给 ActivityMain，并且运行 ActivityMain.java 中的 onActivityResult()里的代码。

⑦ 第 28～33 行完成从 ActivityMain 中的 intent1 对象中取出其附加信息，并赋予 extras (见第 29 行)。如果其键-值对非空，就取出名为“activitymain”的对应值给字符串变量 data，并将 data 的值显示在标题栏信息中。

(6) 编写根目录下的 AndroidManifest.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
      package = "cn.com.sgmsc.Activity_Intent"
      android:versionCode = "1"
      android:versionName = "1.0">
    <application android:icon = "@drawable/icon" android:label = "@string/app_name">
        <activity android:name = ".ActivityMain"
                  android:label = "@string/app_name">
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </activity>

      <activity android:name = ".ActivitySub"></activity>
```

```
16
17     </application>
18     <uses-sdk android:minSdkVersion="10" />
19
20 </manifest>
```

第 15 行增加对 Activity 组件 ActivitySub 的声明。这个声明仅声明了 Activity 的名称，没有其他信息。

【运行结果】 在 Eclipse 中启动 Android 模拟器，注意，首次启动 Android 模拟器需要时间较长，建议在一打开 Eclipse 时就将 Android 模拟器打开，然后可以去做些别的工作。这样，在需要运行 Android 应用程序时可以节省些时间。然后运行 Activity_Intent 项目。运行结果如图 3-3～图 3-5 所示。



图 3-3　首次进入时的 ActivityMain 界面



图 3-4　单击按钮后进入 ActivitySub 界面



图 3-5　单击按钮后返回 ActivityMain 界面

在案例 3.1 中，多次提到了 Bundle 对象。在 Android 中，关于 Activity 之间的消息的传递经常会用到 Bundle。因为在多个 Activity 之间跳转时，使用 Bundle 可以比较方便保存并发送一组字符串信息，这个字符串信息可以是一些消息，也可以是用户界面的某个状态信息等。

3.5　用户界面状态保存

在 Android 中，一个 Activity 被激活并运行，实际上是建立了一个 Activity 的实例。当 Activity 实例与用户交互时，会产生一些状态信息。如用户所选取的值、光标的位置等。如果此时该 Activity 实例进入“暂停”或“停止”状态时，需要保存这些临时的状态信息。这是 Android 应用程序经常会遇到的问题。通常情况下，Android 使用 SharedPreferences 对象或 Bundle 对象来保存这些状态信息。

3.5.1　使用 SharedPreferences 对象

SharedPreferences 是 Android 平台上一个轻量级的存储类，主要是保存一些 Activity 的状态信息如一些用户个性化设置的字体、颜色、位置等参数信息。在 Activity 中重载窗口状态的参数 SaveInstanceState 一般使用 SharedPreferences 完成保存。

通过名称顾名思义，SharedPreferences 好像是一个共享的 Preferences，但其共享的范围局限在同一个 Package 中。换句话说，SharedPreferences 是将数据保存在本应用项目的私有

存储区内，这些存储区的数据只能被本应用项目的程序所读取。所以实际上，对于外部的应用程序而言，SharedPreferences 对象的访问权限是私有的。

在所有的 Android SDK 存储技术中，SharedPreferences 技术是最容易理解和使用的，因为它处理的就是一个键-值对的二元组，并保存在一个内部的 XML 文件中，可以在 Eclipse 的 DDMS 中的 File Explorer 中的/data/data//shares _ prefs 下找到这个 XML 文件。SharedPreferences 支持常见的 String、Long、Float、Integer、Boolean 等数据类型。

每个 Activity 都有一个匿名的 SharedPreferences 对象。获取 Activity 的匿名 SharedPreferences 对象，并对它进行读、写操作，本书将用个小例子作简单介绍，其详细的使用方法参见第 6 章。

例如想将窗口中一个 TextView 控件中的信息保存到 Activity 的匿名 SharedPreferences 中，可以使用下列代码片断实现。

```
1  Protected void saveActivityPreferences(){
2      SharedPreferences activityPref = getPreferences(Activity.MODE_PRIVATE);
3              //获取 Activity 的匿名 SharedPreferrences 对象
4      Editor editor = activityPref.edit();                          //获取对象的编辑器
5      TextView textView = (TextView)findViewById(R.id.textView);    //获取 TextView 控件对象
6      editor.putString("TextValue",textView.getText().toString()); //存储 TextView 控件信息
7      editor.commit();                                              //提交并保存
8  }
```

(1) 第 4 行是得到一个 SharedPreferences 对象的编辑器 editor，以便让要保存的信息写到这个对象中。

(2) 第 6 行是将窗口中的 TextView 控件中的内容写入 editor，其中，“TextValue”是键-值对中的名，使用 textView. getText(). toString()方法获得的字符串信息则是键-值对中的值。

从这段代码中可以明白，如果要用 Activity 的 SharedPreferences 对象保存信息，在编写代码时一般有如下步骤。

(1) 获得 SharedPreferences 对象。

(2) 获得 SharedPreferences. Editor 对象。

(3) 使用 put...()方法保存键-值对。例如保存字符串型使用 putString()方法。

(4) commit()方法进行提交。

3.5.2 使用 Bundle 对象

在 Activity 生命周期的多个方法中，都使用了 Bundle 对象来保存 Activity 相关的状态信息。如方法 onCreate()，onSaveInstanceState()，onRestoreInstanceState()。

Bundle 对象封装的数据也是键-值对的二元组，用 Bundle 绑定数据便于数据处理。所以，利用它的数据封装能力，将要传递的数据或参数通过 Intent 对象来传递到不同的 Activity 之间，是比较简便的办法。Bundle 类继承自 Android. os. Bundle 类。每一个 Android 的代码程序都需要使用它，所以在每个类定义的代码文件前部都必须加上“import android. os. Bundle;”。

在 Android 应用程序的运行过程中，Bundle 对象中的数据是保存在应用程序的上下文中的，如果应用程序退出，相应的上下文也就销毁了，自然，Bundle 对象也就不存在了。

使用 Bundle 对象的常用方法如表 3-5 所示。

表 3-5 Bundle 对象的常用方法说明

方　法	返　回　值	描　述
clear()		清除所有保存的数据
clone()	Bundle 对象	克隆当前 Bundle 对象
get(String key)	指定 key 所映射的值	返回指定键所映射的值。如果对于此键来说，映射不包含任何映射关系，则返回 null
getString(String key)	指定 key 的 String 类型的值	返回指定键所映射的值，如果在映射中不存在由指定键到 String 类型值的映射关系或者指定键在映射关系中显式映射到 null，则返回 null
isEmpty()	true 或 false	如果 Bundle 对象中不包含任何键-值映射关系，则返回 true，否则返回 false
putString(String key, String value)		给指定的键关联一个指定的 String 类型值。如果此键以前包含一个映射关系，则旧值被替换。键和值都可以为 null
remove(String key)		如果 Bundle 对象中存在该键的映射关系，则将其删除

在案例 3.1 的 ActivityMain.java 和 ActivitySub.java 中都有使用 getString()和 putString()方法。请读者回去看看代码的用法，在此不作赘述。

3.5.3 SharedPreferences 与 Bundle 的区别

SharedPreferences 是简单的存储持久化的设置，就像用户每次打开应用程序时的主页，它只是一些简单的键-值对来操作。它将数据保存在一个 XML 文件中。常用于保存应用程序结束前的一些界面状态信息，以便下次再打开应用程序时能恢复关闭前的窗口画面。

Bundle 是将数据传递到另一个上下文中或保存，或回复你自己状态的数据存储方式。它的数据不是持久化状态。常用于应用程序的运行时，几个 Activity 相互跳转时，保存前一个 Activity 的界面状态信息，或用于几个不同的 Activity 之间的数据传递等。

小结

本章主要介绍了 Android 应用程序运行时的控制和通信。由于在应用程序运行时直接与用户交互的是 Activity 组件，所以本章以 Activity 组件为重点，逐步介绍了 Android 系统是如何使用任务、进程和线程来协调多个 Activity 在其生命周期中的各种状态的，以及每个 Activity 在其生命周期中从一个状态转变为另一个状态时，系统是如何自动有序地调用生命周期方法的。介绍了多个 Activity 之间相互跳转时，是如何传递消息，如何保存相关的状态信息的。

第 2 章以静态的方式介绍 Android 应用程序的组成部分，第 3 章以动态的方式介绍 Android 应用程序的内部控制通信机制。虽然这两章中对某些知识点有些重复提及，但侧重点不同，所阐述的内容也不相同。通过这一静一动的介绍，相信读者可以较全面地了解了

Android 应用程序的框架结构及运行机制，但是如何为这个框架赋予实际应用价值，还需要很多构件来充实它。从第 4 章起，将逐渐学习并使用它们。



练习

阅读 Android SDK 例程中，多个 Activity 之间的转换调用。例程路径：SDK 文件夹下的 samples 子文件夹中，本书是 E:\android-sdk\samples。

第4章 Android常用基本控件

控件是 Android 用户界面中的一个个组成元素，在介绍它们之前，读者必须了解所有控件的父类 View（视图），它好比一个盛放控件的容器。

4.1 View 类概述

对于一个 Android 应用来说，android. app. Activity 类实例是一个最基本的功能单元。一个 Activity 实例可以做很多的事情，但是它本身无法显示在屏幕上，而是借助于 View 和 ViewGroup，它们是 Android 平台上最基本的两个用户界面表达单元。

4.1.1 关于 View

View 对象是一个数据体，它的属性存储了用于屏幕上一块矩形区域的布局参数及内容，并负责这块它所辖的矩形区域之中所有测量、布局、焦点转换、卷动以及按键/触摸手势的处理。作为一个用户界面对象，View 同时也担任着用户交互关键点以及交互事件接受者的角色。View 来自 android. view. View 类。在屏幕上，每个 View 的子类对象都是 android. view. View 类的一个实例，所有在 Activity 中要用到可视化的窗体控件，就必须在该类定义的代码文件前部加上“import android. view. View；”。

View 类是所有屏幕表达单元的基类。View 类的所有属性，以及在 View 类中定义的所有成员方法，其子类都全部继承。表 4-1 描述了 View 类常用的属性及其对应方法。

表 4-1 View 类常用的属性及对应方法说明

属　性	方　法	描　述
android：background	setBackgroundResource(int)	设置背景
android：clickable	setClickable(boolean)	设置 View 是否响应单击事件
android：visible	setVisible(int)	控制 View 的可见性
android：focusable	setFocusable(boolean)	控制 View 是否可以获取焦点
android：id	setId(int)	为 View 设置标识符，可通过 findViewById() 方法获取
android：longClickable	setLongClickable(boolean)	设置 View 是否响应长单击事件
android：saveEnabled	setSaveEnabled(boolean)	如果未作设置，当 View 被冻结时将不会保存其状态

4.1.2 关于 ViewGroup

ViewGroup 是一个特殊的 View 类，它继承于 android. view. View，是 View 类的子类。ViewGroup 是充当其他控件的容器，在其内可以有 View 对象和 ViewGroup 对象，它的功能就是装载和管理下一层的 View 对象和 ViewGroup 对象。确切地说，它是所有容器类的基类。

View 的布局显示方式直接影响用户界面，准确地说是一个 ViewGroup 中包含的一些 View 对象是如何布局的。在 ViewGroup 中，定义了一个嵌套类 ViewGroup. LayoutParams。这个类定义了一个显示对象的位置、大小等属性，View 通过 LayoutParams 中的这些属性值来告诉 Android 系统，它们将如何放置。这就是布局类。

在 Android 中，布局类通常都在布局的 XML 文件中描述，而很少在 Java 代码中动态定义。如果使用 Java 代码定义布局，那么在代码文件的前部必须加上“import android. view. ViewGroup;”。

4.2 常见布局

布局(Layout)是 ViewGroup 的子类，为视图控件提供排列结构。Android 提供了一些预定义的视图组，其中包括 FrameLayout，LinearLayout，TableLayout，RelativeLayout，AbsoluteLayout。每个都为定义子视图和布局结构提供了一套独特的布局参数。下面分别介绍。

4.2.1 帧布局

帧布局(FrameLayout)是最简单的一个布局对象。它里面只显示一个 View 对象。Android 屏幕元素中所有的显示对象都将会固定在屏幕的左上角，不能指定位置。如果在帧布局中有多个显示对象，那么后一个将会直接在前一个之上进行覆盖显示，把前一个部分或全部遮住。当然，如果后一个显示对象是透明的，则前一个对象会显现出来。使用帧布局设计的效果如图 4-1 所示。

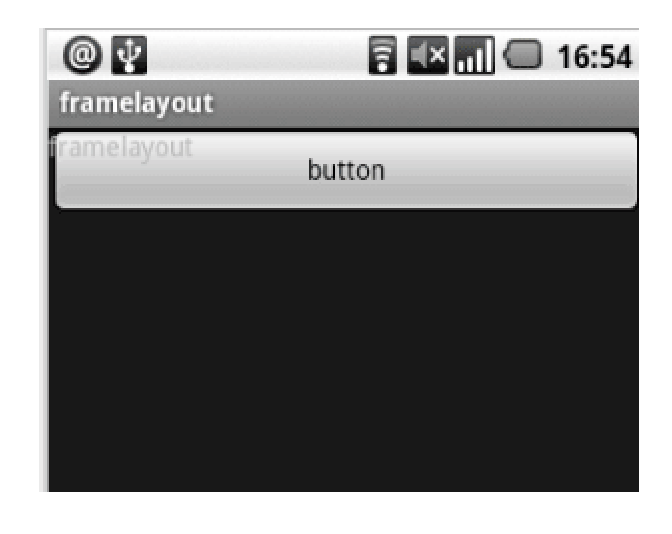


图 4-1 帧布局的显示效果

4.2.2 线性布局

线性布局(LinearLayout)是最常用的布局方式。它以单一方向对其中的 View 对象进行排列显示，如果以垂直排列显示，则屏幕布局中将只有一列；如果以水平排列显示，则屏幕布局中将只有一行。对于多个显示对象，线性布局保持它们之间的间隔以及互相对齐。使用线性布局设计的效果如图 4-2 和图 4-3 所示。

在进行线性布局中，可以通过设置其属性值来改变排列在其中的显示对象的显示效果。常用的属性如表 4-2 所示。



图 4-2 横向线性布局显示效果



图 4-3 纵向线性布局显示的效果

表 4-2 线性布局常用属性说明

属　性	说　明
android：background	设置背景颜色
android：orientation	设置线性布局的排列方向。Horizontal 表示横向，vertical 表示纵向
android：gravity	设置线性布局内部显示对象的位置对齐布局方式
android：layout_width	设置线性布局的宽度。fill_parent 表示填充整个屏幕，wrap_content 表示按对象上的文字的宽度不同而确定显示对象的宽度
android：layout_height	设置线性布局的高度。属性值同 android：layout_width
android：layout_weight	设置线性布局内部多个显示对象的重要度赋值，按比例为它们划分空间

这里，layout_width、layout_height 及其取值同样适用于其他类型的布局。gravity 是一个很有用的属性，它用来设置线性布局内显示对象的对齐方式。gravity 可取的值及其说明如表 4-3 所示。

表 4-3 gravity 可取的属性值及说明

属 性 值	说　明
top	不改变显示对象的大小，对齐到窗口顶部
bottom	不改变显示对象的大小，对齐到窗口底部
left	不改变显示对象的大小，对齐到窗口左侧
right	不改变显示对象的大小，对齐到窗口右侧
center_vertical	不改变显示对象的大小，对齐到容器纵向中央位置
center_horizontal	不改变显示对象的大小，对齐到容器横向中央位置
center	不改变显示对象的大小，对齐到容器中央位置
fill_vertical	若有可能，纵向拉伸以填满容器
fill_horizontal	若有可能，横向拉伸以填满容器
fill	若有可能，纵向、横向同时拉伸以填满容器

在线性布局中，如果有多个 View 对象，那么所有的对象都有一个 layout_weight 值，默认为 0，其意思是按显示对象的原来大小显示在屏幕上。如果 layout_weight 值大于 0，则将本线性布局的父视图中的可用空间分割，分割大小具体取决于每一个显示对象的 layout_weight 值。例如，如果在水平的线性布局中有一个文本标签和两个文本编辑对象，如果该文本标签并无指定 layout_weight 值，那么它将占据需要提供的最少空间；如果两个文本编辑对象每一个的 layout_weight 值都设置为 1，则两者平分布局区域的剩余宽度，因为这里声明两者的重要度相等；如果两个文本编辑对象中第一个 layout_weight 值设置为 1，而第二个的值设置为 2，于是，第一个占剩余宽度的三分之二，而第二个占三分之一的宽度，因为 layout_weight 的数值越小，越重要。可见，当线性布局中有多个显示对象时，layout_weight 属性很重要，它可以避免在一个大屏幕中，一串小对象挤成一堆的情况。

【案例 4.1】 设计出如图 4-2 所示的布局文件。

【说明】 图 4-2 中有 5 个按钮，它们以横向的线性排列方式显示在屏幕的顶部。因此在布局文件中要使用 layout_weight 属性且值都相同。

【代码】 在 Eclipse 中导入 Activity_LinearLayout 项目(可以从清华大学出版社为本书提供的下载资源的源代码中获得)。打开 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
              android:layout_width = "fill_parent"
              android:layout_height = "fill_parent"
              android:orientation = "horizontal">
  <Button android:id = "@ + id/button1"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:text = "Hello, I am a Button1"
              android:layout_weight = "1"
  />
      <Button android:id = "@ + id/button2"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:text = "Hello, I am a Button2"
              android:layout_weight = "1"
    />
      <Button android:id = "@ + id/button3"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:text = "Hello, I am a Button3"
              android:layout_weight = "1"
  />
      <Button android:id = "@ + id/button4"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:text = "Hello, I am a Button4"
              android:layout_weight = "1"
    />
      <Button android:id = "@ + id/button5"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:text = "Hello, I am a Button5"
              android:layout_weight = "1"
```

```
35      />
36 </LinearLayout>
```

(1) 第 2～36 行声明一个线性布局。其中，线性布局的属性定义在第 2～5 行，第 2 行是每个 Android 布局文件必有的；第 3～4 行设置线性布局的宽和高属性值，值都是 fill_parent，即充满容器的宽和高；第 5 行设置线性布局的方向是 horizontal，即是横向的。

(2) 第 6～11 行、第 12～17 行、第 18～23 行、第 24～29 行、第 30～35 行分别声明了 5 个按钮，每个按钮中的文本都比较长，如果不设置 layout_weight 的值，在屏幕的一行中是显示不下所有 5 个按钮的。

思考一下：

如果将每个 Button 的属性"android:layout_weight＝1"都删掉，会出现什么样的显示效果？

线性布局允许嵌套。在开发中，往往在一个横向的线性布局里嵌套一个纵向的线性布局，或在一个纵向的线性布局里嵌套一个横向的线性布局。这是非常常用的技巧。

【案例 4.2】 设计出如图 4-4 所示的布局文件。

【说明】 从图 4-4 中可以看到，这个布局应该是：外层是一个纵向的线性布局，并且上下两部分的高度是不等比的。内层是一个横向的线性布局，并且是等分的两部分。有 5 个按钮，它们以横向的线性排列方式显示在屏幕的顶部。因此在布局文件中要使用 layout_weight 属性。

【代码】 在 Eclipse 中导入 Activity_NestLineLayout 项目(可以从本教材的源代码中获得)。打开 res/layout 目录下的 main.xml 文件，代码如下所示。

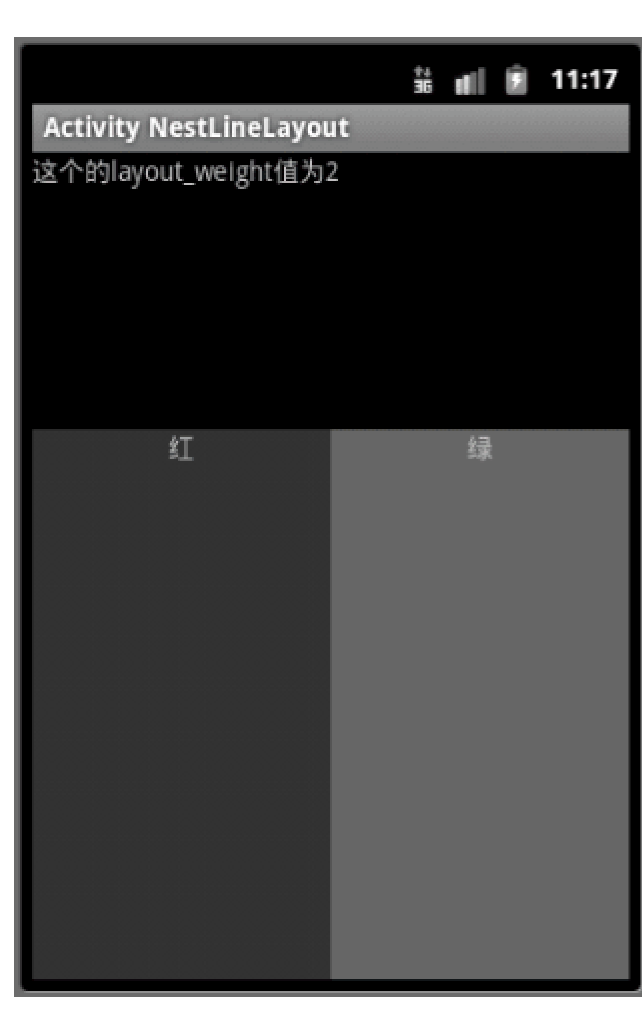


图 4-4 嵌套线性布局显示的效果

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
   <LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="2">
      <TextView
          android:text="这个的 layout_weight 值为 2"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
       />
    </LinearLayout>
    <LinearLayout
       android:orientation="horizontal"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:layout_weight="1">
       <TextView
          android:text="红"
```

```
24             android:gravity = "center_horizontal"
25             android:background = "#aa0000"
26             android:layout_width = "wrap_content"
27             android:layout_height = "fill_parent"
28             android:layout_weight = "1"/>
29         <TextView
30             android:text = "绿"
31             android:gravity = "center_horizontal"
32             android:background = "#00aa00"
33             android:layout_width = "wrap_content"
34             android:layout_height = "fill_parent"
35             android:layout_weight = "1"/>
36     </LinearLayout>
37 </LinearLayout>
```

(1) 第 2～37 行声明的是外层线性布局，从第 3 行知此线性布局的方向是 vertical，即是纵向的。

(2) 第 6～16 行声明的是上面的内层线性布局。为了与下面的高度不一致，设置此线性布局的 layout_weight 属性值为 2。由于此线性布局内只有一个＜TextView＞控件，所以此线性布局的方向可为横向，也可为纵向，这里设置的是纵向。

(3) 第 17～36 行声明的是下面的内层线性布局。此线性布局内有两个＜TextView＞控件，且其高度应该是上面的两倍高，所有设置此线性布局的 layout_weight 属性值为 1；设置此线性布局的方向是 horizontal，即是横向的。

4.2.3 表格布局

表格布局(TableLayout)是线性布局的特殊类。以拥有任意行列的表格对 View 对象进行布局，每个单元格内只显示一个 View 对象，但单元格的边框线不可见。表格布局类每一行为一个 TableRow 对象，在 TableRow 中可以添加显示对象，每添加一个显示对象为一列。

在表格布局中，一个列的宽度由该列中最宽的那个单元格确定，而表格的宽度则是由父容器确定。布局中可以有空的单元格，也可以像 HTML 文件一样，一个单元格可以跨越多个列。

TableLayout 继承自 LinearLayout 类，它继承了线性布局所拥有的属性和方法，但表格布局还有它特有的，为列设置有属性和方法。表 4-4 是对表格布局列的常用属性的描述。

表 4-4 表格布局列的常用属性说明

属性	说明
android:collapseColumns	设置指定列为 Collapsed，列号从 0 开始计算。如果列被标识为 Collapsed，则该列将会被隐藏
android:shrinkColumns	设置指定列为 Shrinkable，列号从 0 开始计算。如果列被标识为 Shrinkable，则该列的宽度可以进行收缩，以使表格能够适应其父容器的大小
android:stretchColumns	设置指定列为 Stretchable，列号从 0 开始计算。如果列被标识为 Stretchable，则该列的宽度可以进行拉伸，以填满表格中空闲的空间

注意，一个列可以同时具有 Shrinkable 和 Stretchable 属性，在这种情况下，该列的宽度将可以任意地进行收缩或拉伸以适应父容器。

【案例 4.3】 设计一个三行三列的表格布局，其中第一行前两列分别放两个按钮，第二行后两列分别放两个按钮，第三行第三列放一个按钮。

【说明】 这个表格布局每一行的显示对象的位置都不相同，在布局文件中需要使用 TableRow 类来逐行定义。另外，这里没有对每个按钮的宽度作要求，在此默认是等宽的。

【代码】 在 Eclipse 中导入 Activity_TableLayout 项目（可以从本教材的源代码中获得）。打开 res/layout 目录下的 main.xml 文件，代码如下所示。

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:layout_width="fill_parent"
4                 android:layout_height="fill_parent"
5                 android:shrinkColumns="0,1,2">
6     <TableRow><!-- row1 -->
7         <Button android:id="@+id/button1"
8             android:layout_width="wrap_content"
9             android:layout_height="wrap_content"
10            android:text="Hello, I am a Button1"
11            android:layout_column="0"
12        />
13      <Button android:id="@+id/button2"
14            android:layout_width="wrap_content"
15            android:layout_height="wrap_content"
16            android:text="Hello, I am a Button2"
17            android:layout_column="1"
18         />
19     </TableRow>
20     <TableRow><!-- row2 -->
21         <Button android:id="@+id/button3"
22            android:layout_width="wrap_content"
23            android:layout_height="wrap_content"
24            android:text="Hello, I am a Button3"
25            android:layout_column="1"
26          />
27         <Button android:id="@+id/button4"
28            android:layout_width="wrap_content"
29            android:layout_height="wrap_content"
30            android:text="Hello, I am a Button4"
31            android:layout_column="1"
32        />
33    </TableRow>
34    <TableRow><!-- row3 -->
35       <Button android:id="@+id/button5"
36            android:layout_width="wrap_content"
37            android:layout_height="wrap_content"
38            android:text="Hello, I am a Button5"
39            android:layout_column="2"
40        />
41    </TableRow>
42 </TableLayout>
```

（1）第 2～42 行声明的是表格布局。其中第 5 行定义了该表格是三列的，且是可收缩的，这样，布局按照父容器的宽度平分三列宽度，如果每列中的显示对象超出了单元格的宽度就自动收缩至单元格宽度。列号从 0 开始计算。

（2）第 6～19 行声明了表格的第一行。其中有两个按钮<Button>标签，尽管每个按钮

的 layout_width 和 layout_height 值都是 wrap_content，即以文本的宽度、高度为按钮的尺寸，但是由于在<TableLayout>中设置属性"android:shrinkColumns="0,1,2""，所以最终三列还是以三等份来平分父容器的宽度。

(3) 第 11 行表示在此行第一列显示按钮 1，第 17 行表示在此行第二列显示按钮 2。

(4) 第 20～33 行声明了表格的第二行，其中定义了两个<Button>标签。

(5) 第 25 行表示在此行第二列显示按钮 3，第 31 行表示在此行第二列之后添加一列(即第三列)显示按钮 4。

(6) 第 34～41 行声明了表格的第三行，其中定义了一个<Button>标签，并定义在第三列显示。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_TableLayout 项目。运行结果如图 4-5 所示。

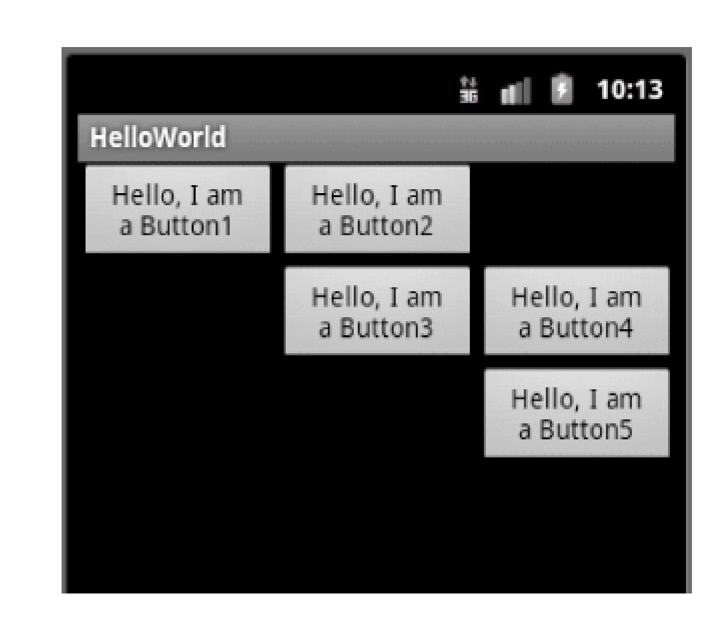


图 4-5 三行三列表格布局的显示效果

4.2.4 相对布局

相对布局(RelativeLayout)允许通过指定 View 对象相对于其他显示对象或父级对象的相对位置来布局。如一个按钮可以放于另一个按钮的右边，或者可以放在布局管理器的中央等。注意，如果对象 A 的位置是相对于对象 B 来决定的，那么必须先定义了对象 B，然后才能定义对象 A。

在进行相对布局时用到的属性较多，下面按其属性值的归类分别进行介绍。

属性值为 true 或 false 的属性如表 4-5 所示。

表 4-5 相对布局中取值为 true 或 false 的属性及说明

属 性	说 明
android:layout_centerHorizontal	当前显示对象位于父控件的横向中间位置
android:layout_centerVertical	当前显示对象位于父控件的纵向中间位置
android: layout_centerInParent	当前显示对象位于父控件的中央位置
android: layout_alignParentBottom	当前显示对象底端与父控件的底端对齐
android: layout_alignParentLeft	当前显示对象左侧与父控件的左侧对齐
android: layout_alignParentRight	当前显示对象右侧与父控件的右侧对齐
android: layout_alignParentTop	当前显示对象顶端与父控件的顶端对齐
android: layout_alignWithParentIfMissing	参照对象不存在或不可见时参照父控件

属性值为其他控件的 id 的属性如表 4-6 所示。

表 4-6 相对布局中取值为其他控件 id 的属性及说明

属 性	说 明
android:layout_toRightOf	使当前显示对象位于给出 id 控件的右侧
android:layout_toLeftOf	使当前显示对象位于给出 id 控件的左侧
android: layout_above	使当前显示对象位于给出 id 控件的上方
android: layout_below	使当前显示对象位于给出 id 控件的下方
android: layout_alignTop	使当前显示对象的上边界与给出 id 控件的上边界对齐
android: layout_alignBottom	使当前显示对象的下边界与给出 id 控件的下边界对齐
android: layout_alignLeft	使当前显示对象的左边界与给出 id 控件的左边界对齐
android: layout_alignRight	使当前显示对象的右边界与给出 id 控件的右边界对齐

属性值为像素单位的属性如表 4-7 所示。

表 4-7　相对布局中取值为像素单位的属性及说明

属　　性	说　　明
android:layout_marginLeft	当前显示对象左侧的留白
android:layout_marginRight	当前显示对象右侧的留白
android: layout_marginTop	当前显示对象上方的留白
android: layout_marginBottom	当前显示对象下方的留白

注意，在进行相对布局时要避免循环依赖。例如，在相对布局中设置了父容器的排列方式为 WRAP_CONTENT，即表示它将以容器内的显示子对象的大小为尺寸，如果此时将其子对象设置为 ALIGN_PARENT_BOTTOM，即表示它将与父容器的底部对齐。这就出现了循环依赖的排列，因此造成子控件与父控件相互依赖和参照的错误。

【案例 4.4】 试使用相对布局来设计如图 4-6 所示的界面效果。

【说明】 通常可以使用嵌套的线性布局或者使用表格布局来实现如图 4-6 所示的用户界面，但在此需要使用相对布局实现。先来分析一下界面中各个对象的位置，最上方是一个与屏幕左对齐的文字串，可使用＜TextView＞控件；它的下方是一个可输入的编辑框，并充满整个屏幕宽度，可使用＜EditText＞控件；它的下方是两个并排的按钮，并且是与屏幕右对齐的。当设计它们的显示位置时，一定要找准参照对象。



图 4-6　一个简单的信息录入框界面

【代码】 在 Eclipse 中导入 Activity_RelativeLayout 项目（可以从本教材的源代码中获得）。打开 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent">

        <TextView
            android:id = "@ + id/label"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "Type here:"
        />
        <EditText
            android:id = "@ + id/entry"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:background = "@android:drawable/editbox_background"
            android:layout_below = "@id/label"/>
        <Button
            android:id = "@ + id/ok"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_below = "@id/entry"
            android:layout_alignParentRight = "true"
```

```
24              android:layout_marginLeft = "10dip"
25              android:text = "OK" />
26          < Button
27              android:layout_width = "wrap_content"
28              android:layout_height = "wrap_content"
29              android:layout_toLeftOf = "@id/ok"
30              android:layout_alignTop = "@id/ok"
31              android:text = "Cancel" />
32
33 </RelativeLayout>
```

(1) 第 2～33 行声明了一个相对布局，并且这个相对布局的宽度和高度都充满屏幕。

(2) 第 6～11 行声明了一个<TextView>控件，它的 ID 为“label”，显示内容为“Type here:”。第 8 行设置该文本框的宽度是 fill_parent，即填满其父容器。第 9 行设置的高度为 wrap_content，即其高度为文本的高度。

(3) 第 12～17 行声明了一个<EditText>控件，它的 ID 为“entry”。第 17 行设置本编辑框的位置，指定它位于 ID 为“label”的<TextView>控件的下方。

(4) 第 18～25 行声明了一个<Botton>控件，它的 ID 为“ok”。第 22 行设置本按钮的位置，指定它位于 ID 为“entry”的< EditText >控件的下方；第 23 行指定它与父容器的右侧对齐；第 24 行指定本按钮的左侧留白为 10dip；按钮上显示的文本为“OK”。

(5) 第 26～31 行声明了一个<Botton>控件。第 29 行设置本按钮位于 ID 为“ok”的按钮控件的左侧，并由第 24 行知与“ok”按钮控件的间距为 10dip；第 30 行指定它与 OK 按钮控件的上边界对齐；按钮上显示的文本为“Cancel”。

4.2.5 绝对布局

绝对布局(AbsoluteLayout)允许以坐标的方式，指定 View 对象的具体位置，左上角的坐标为(0，0)，向下及向右，坐标值变大。这种布局管理器由于显示对象的位置定死了，所以在不同的设备上，有可能会出现最终的显示效果不一致的情况。

由于每一个显示对象都是通过计算其坐标来定位和布局的，与周围的其他控件和父容器无关，所以绝对布局是一种用起来比较费时的布局管理器。

现在已经介绍完 Android 的 5 类布局管理器。这些布局各有特点，在开发时，应该根据实际应用的用户界面，选择合适的布局管理器。下面给出一些应用实例的用户界面，如图 4-7 所示。你能说出它们各使用了什么布局吗？



图 4-7 几种常见的用户界面布局

4.3 Android 常见的基本控件

部件(Widget)是为构建用户交互界面提供服务的视图对象。Android 和其他开发语言一样,提供一套完整的部件实现。常用的 Widget 包括 TextView、EditText、Button、RadioButton、Checkbox 和 ScrollView 等。由于它们每一个都可以与用户进行交互,所以把 Widget 类中的每一个部件称为控件。

Widget 类是 View 类的子类,Widget 内的所有控件都继承了 View 类中的属性和方法。可以在 android.widget 包中找到 Android 提供的控件列表。下面分别进行介绍。

4.3.1 TextView

TextView(文本框)是用来向用户显示文本内容的控件。它在 android.widget.TextView 包中定义,如果在 Java 类设计中使用它,则需要在相应的代码文件前部加上"import android.widget.TextView;"。

TextView 控件中有很多可以在 XML 文件中设置的属性,这些属性同样可以在代码中以方法动态声明。常用的属性和对应的方法如表 4-8 所示。

表 4-8 TextView 常用的属性和对应方法说明

属 性	方 法	描 述
android:gravity	setGravity(int)	设置 TextView 在 x 轴和 y 轴方向上的显示方式
android:height	setHeight(int)	设置 TextView 的高度,以像素为单位
android:width	setWidth(int)	设置 TextView 的宽度,以像素为单位
android:hint	setHint(int)	当 TextView 中显示的文本内容为空时,显示的文本提示信息
android:padding	setPadding(int)	设置 TextView 中显示文本与其父容器边界的间距
android:text	setText(CharSequence)	为 TextView 设置显示的文本内容
android:textColor	setTextColor(ColorStateList)	设置 TextView 的文本颜色
android:typeface	setTypeface(Typeface)	设置 TextView 的文本字体
android:textSize	setTextSize(float)	设置 TextView 的文本大小
android:textStyle	setTextStyle(TextStyle)	设置 TextView 的文本字形,如粗体、斜体等

在使用中,要注意一些属性的区别。

1. android:gravity 与 android:layout_gravity 的区别

android:gravity 用于设置这个 View 内的所有子元素的对齐方式,而 android:layout_gravity 用于设置这个 View 在其父容器中的对齐方式。

2. android:padding 与 android:layout_margin 的区别

padding 是站在父 View 的角度描述问题的,它规定它里面的内容必须与这个父 View 边界的距离。而 margin 则是站在自己的角度描述问题的,规定自己和其他(上下左右)的 View 之间的距离,如果同一级只有一个 View,那么它的效果基本上就和 padding 一样了。

【案例 4.5】 试设计如图 4-8 所示的布局文件片段。

【说明】 这个 TextView 控件有背景色，需要使用 android：background 属性，文本颜色是白色的，需要使用 android：textColor 属性。颜色常量使用"#RRGGBB"6 位十六进制数表示。

【代码】 布局文件中的代码片断如下所示。



图 4-8　一个 TextView 控件

```
1  <TextView
2      android:id = "@ + id/text_view"
3      android:layout_width = "fill_parent"
4      android:layout_height = "wrap_content"
5      android:textSize = "16sp"
6      android:textColor = "#ffffff"
7      android:padding = "10dip"
8      android:background = "#cc0000"
9      android:text = "这里是 TextView,你可以在这里输入需要显示的文字信息.."
10 />
```

（1）第 1～10 行声明了一个 TextView 控件。

（2）第 5 行设置文本的字符大小，即 16sp。在 2.2.1 节中已经说明描述字符的单位使用 sp。

（3）第 6 行设置文本的颜色，"#ffffff"为纯白色。

（4）第 7 行设置文本与其父容器(即文本框)边界的间距为 10 像素。可以看出图 4-8 中的文本与红色区域的边界是有 10 个像素的间隔。

（5）第 8 行设置文本框的背景颜色，"#cc0000"为红色。

（6）第 9 行设置文本框的显示内容。

4.3.2　EditText

EditText(编辑框)是用来向用户显示文本内容，并允许用户对文本进行编辑的控件。在开发中经常会使用到这个控件，例如，实现登录界面时，需要用户输入账号、密码等信息，这时需要用到此编辑控件。EditText 控件定义在 android.widget.EditText 包中，如果在 Java 类设计中使用它，则需要在相应的代码文件前部加上"import android.widget.EditText;"。

EditText 类继承自 TextView，有许多 EditText 控件中的属性和对应方法，对 TextView 控件同样适用。下面列出 EditText 控件常用的属性和方法，如表 4-9 所示。

表 4-9　EditText 常用的属性和对应方法说明

属　性	方　法	描　述
android:cursorVisible	setCursorVisible (boolean)	设置光标是否可见，默认可见
android:lines	setLines(int)	设置固定行数以确定 EditText 的高度
android:maxLines	setMaxLines(int)	设置最大的行数
android:maxLength	setFilters(InputFilter)	设置最大的显示长度
android:password	setTransformationMethod (TransformationMethod)	设置显示是否为密码模式
android:phoneNumber	setKeyListener(KeyListener)	设置显示文本只能是电话号码
android:scrollHorizontally	setHorizontallyScrolling(boolean)	设置文本框是否可以进行水平滚动
android:singleLine	setTransformationMethod (TransformationMethod)	设置文本框为单行模式

【案例 4.6】 试设计如图 4-9 所示的布局文件片段。

【说明】 这个 EditText 控件有一定的宽度，并且与屏幕右侧对齐，需要使用 layout_gravity 属性。注意，此时 EditText 控件中的文本是居中的，需要使用 gravity 属性。

【代码】 布局文件中的代码片段如下所示。

```
1  <EditText android:id = "@ + id/edit_text"
2            android:layout_width = "100dip"
3            android:layout_height = "wrap_content"
4            android:text = "one"
5            android:gravity = "center"
6            android:layout_gravity = "right"/>
```

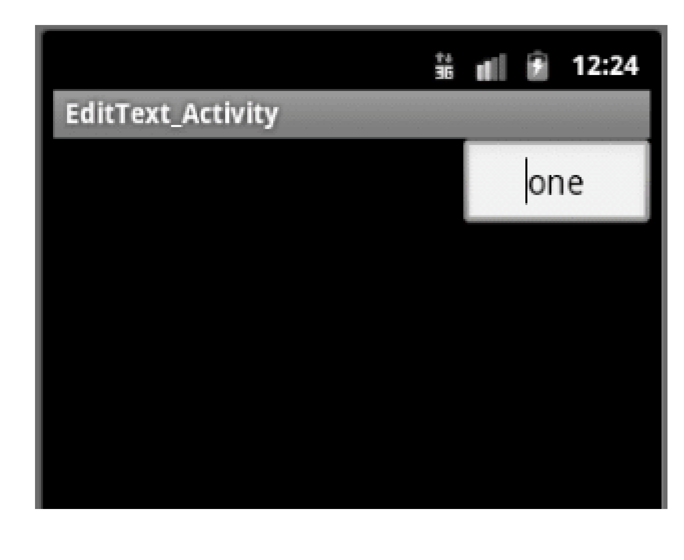


图 4-9 一个 EditText 控件

(1) 第 1～6 行声明了一个 EditText 控件。

(2) 第 2 行设置 EditText 的宽度为 100 像素。

(3) 第 5 行设置 EditText 中的文本居中。

(4) 第 6 行设置 EditText 控件与屏幕右侧对齐。

4.3.3 Button

Button(按钮)是用得最多的控件，在前面的多个示例中都已经见到了按钮。在 Android 平台中，按钮是通过 Button 来实现的。实现过程也非常简单。当用户对按钮按下或单击之后，按钮控件就会触发 onClick 事件，所以需要对按钮设置 setOnClickListener 事件监听。相关按钮的事件及事件监听将在第 5 章作详细介绍。

Button 定义于 android.widget.Button 类中，如果在 Java 类设计中使用它，则需要在相应的代码文件前部加上“import android.widget.Button;”。

Button 继承自 TextView 类，Button 的背景可以是背景颜色，也可以是背景图片，Button 上可显示文本，Button 的大小可设置等，这些属性和相应的方法与 TextView 控件的属性和方法相同。

4.3.4 ImageButton

ImageButton(图片按钮)控件与 Button 控件和功能一致，但在显示效果上是不一样的，它们的主要区别是 ImageButton 中没有 text 属性，即按钮中只显示图片而不显示文本。

ImageButton 继承自 ImageView 类。ImageButton 控件中设置按钮显示的图片可以通过 android:src 属性，如果在代码中动态设置图片则使用 setImageResource(int)方法来实现。

在默认的情况下，ImageButton 与 Button 一样都具有背景色，但是，为了不影响图片的显示，一般要将其背景色设置为透明或其他的图片。

下面通过一个案例来说明如何为 ImageButton 按钮的不同状态设置不同的显示图片。

【案例 4.7】 设计两个 ImageButton 按钮，这两个按钮在未被按下时都显示如图 4-10(a)所示的图片。但当它们被按下时，第一个图片按钮显示不同的背景色，而第二个图片按钮显示如图 4-10(b)所示的图片。


(a)

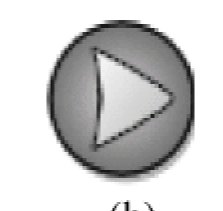

(b)

图 4-10 按钮图片

【说明】 首先必须把这两个图片文件存放在项目的 res 下的 drawable 目录中(也可以是 drawable-hdpi、drawable-ldpi 或 drawable-

mdpi)，然后使用 android:src 属性为 ImageButton 设置显示图片。对于按钮一，只设置了按钮未被按下时显示的图片，对于按钮二，还需要设置按钮被按下时显示的图片，这时，需要在图片资源存放的目录中编写一个 XML 文件，用于描述按钮的两个状态各显示的图片。

【布局设计步骤】

(1) 这里，如图 4-10(a)所示的图片文件名为 play.png，如图 4-10(b)所示的图片文件名为 playb.png。在该项目的 res 目录中创建一个名为 drawable 的目录，将这两个图片文件复制到新建的目录中。

(2) 在 res/layout 目录下创建一个名为 myselector.xml 的文件，用于设置按钮在不同状态下显示不同的图片文件，代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < selector xmlns:android = "http://schemas.android.com/apk/res/android">
3  < item
4          android:state_pressed = "false"
5          android:drawable = "@drawable/play"/>  <! --  设置按钮未被按下时的图片  -->
6  < item
7          android:state_pressed = "true"
8          android:drawable = "@drawable/playb"/> <! --  设置按钮按下时的图片  -->
9  </selector >
```

① 第 2～19 行声明图片按钮的选择标签，使用＜selector＞标签。

② 第 3～5 行声明按钮未被按下状态时的图片资源。这里使用＜item＞标签，按钮未被按下时用“android:state_pressed＝"false"”判断，用属性设置图片源，即“android:drawable＝"@drawable/play"”。

③ 第 6～8 行声明按钮被按下状态时的图片资源，此时设置图片资源为“android:drawable＝"@drawable/playb"”。

(3) 编写 res/layout 目录下的布局文件，在这个项目中设计本案例的布局文件名为 image_button.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical" android:layout_width = "fill_parent"
    android:layout_height = "wrap_content">

    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "两个图片按钮:" />

    < ImageButton
        android:id = "@ + id/image_button1"
        android:src = "@drawable/play"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    < ImageButton
        android:id = "@ + id/image_button2"
        android:src = "@drawable/myselector"
        android:layout_width = "wrap_content"
```

```
20          android:layout_height = "wrap_content"/>
21
22 </LinearLayout>
```

① 第 2～22 行声明了一个线性布局。其内有三个控件，一个 TextView 控件，两个 ImageButton 控件。

② 第 11～15 行声明第一个图片按钮，其 ID 为“image_button1”。第 13 行设置该按钮未被按下时的显示图片为图片资源目录中的“play.png”。

③ 第 16～20 行声明第二个图片按钮，其 ID 为“image_button2”。第 18 行设置该按钮未被按下和被按下两种状态的显示图片由图片资源目录中的 myselector.xml 文件描述。

【运行效果】 图 4-11～图 4-13 分别是这两个图片按钮在两种状态下的运行效果。



图 4-11　两按钮都未被按下时的显示效果



图 4-12　当按钮一被按下时的显示效果

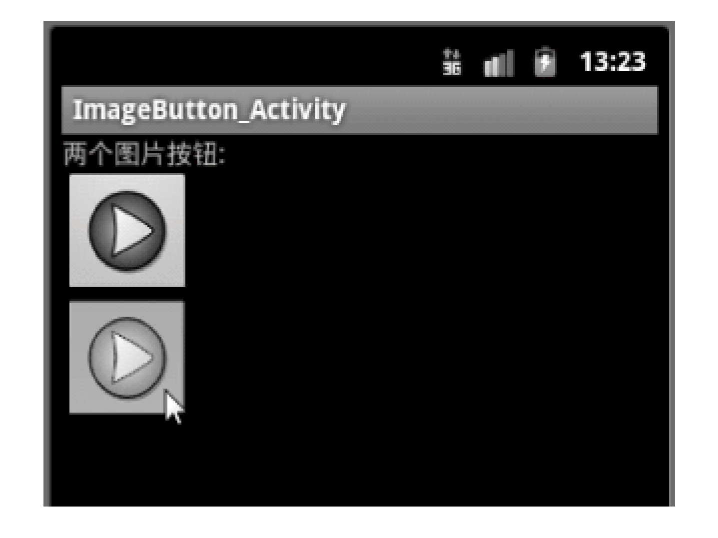


图 4-13　当按钮二被按下时的显示效果

4.3.5　Checkbox 和 RadioButton

CheckBox 与 RadioButton(复选按钮与单选按钮)是开发中经常需要用到的选择按钮，通常一组单选按钮需要编制到一个 RadioGroup 中。相信读者对它们并不陌生。CheckBox 与 RadioButton 都继承自 CompoundButton 类，它们也都从父类 CompoundButton 中继承了一些成员方法，常用的方法及说明如表 4-10 所示。

表 4-10　CheckBox 与 RadioButton 常用的方法及说明

方　　法	描　　述
isChecked ()	判断是否被选中，如果被选中返回 true，否则返回 false
performClick()	调用 OnClickListener 监听器，即模拟一次单击
setChecked(boolean checked)	通过传入的参数设置控件状态
toggle()	置反控件当前的状态，即原为选中状态时返回未选中状态，或原为未选中状态时返回选中状态
setOnCheckedChangeListener(CompoundButton.OnCheckedChangeListener listener)	为控件设置 OnCheckedChangeListener 监听器

【案例 4.8】 试设计如图 4-14 所示的布局文件片段。

【说明】 这组 CheckBox 控件是将文本标签的字形作为复选项，并且是以所见即所得的形式，标注什么字形就显示什么字形，因此用 textStyle 属性。

【代码】　布局文件中的代码片段如下所示。

```
1  <CheckBox android:id = "@ + id/plain_cb"
2          android:text = "Plain"
3          android:layout_width = "wrap_content"
4          android:layout_height = "wrap_content"
5  />
6  <CheckBox android:id = "@ + id/serif_cb"
7          android:text = "Serif"
8          android:layout_width = "wrap_content"
9          android:layout_height = "wrap_content"
10         android:typeface = "serif"
11 />
12 <CheckBox android:id = "@ + id/bold_cb"
13         android:text = "Bold"
14         android:layout_width = "wrap_content"
15         android:layout_height = "wrap_content"
16         android:textStyle = "bold"
17 />
18 <CheckBox android:id  = "@ + id/italic_cb"
19         android:text = "Italic"
20         android:layout_width = "wrap_content"
21         android:layout_height = "wrap_content"
22         android:textStyle = "italic"
23 />
```

(1) 这里声明了 4 个 CheckBox 控件。

(2) 第 10、16、22 行设置 CheckBox 控件显示标签的字形。

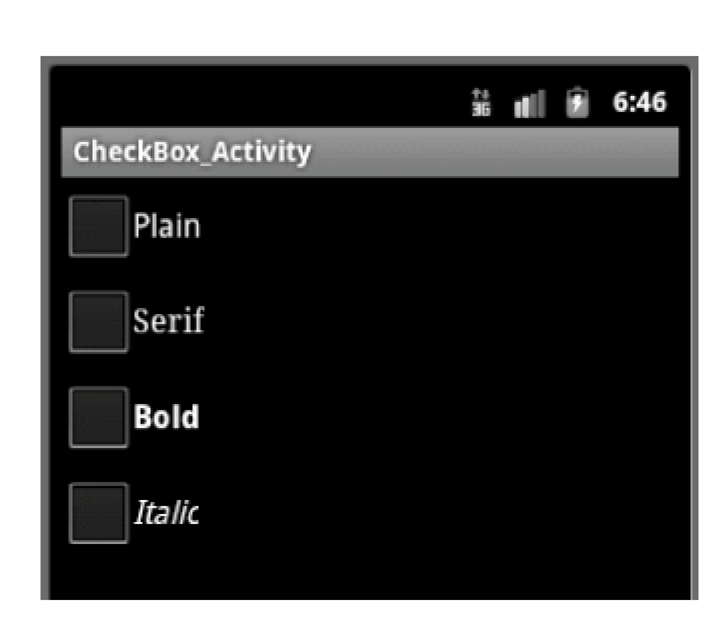


图 4-14　一组 CheckBox 控件

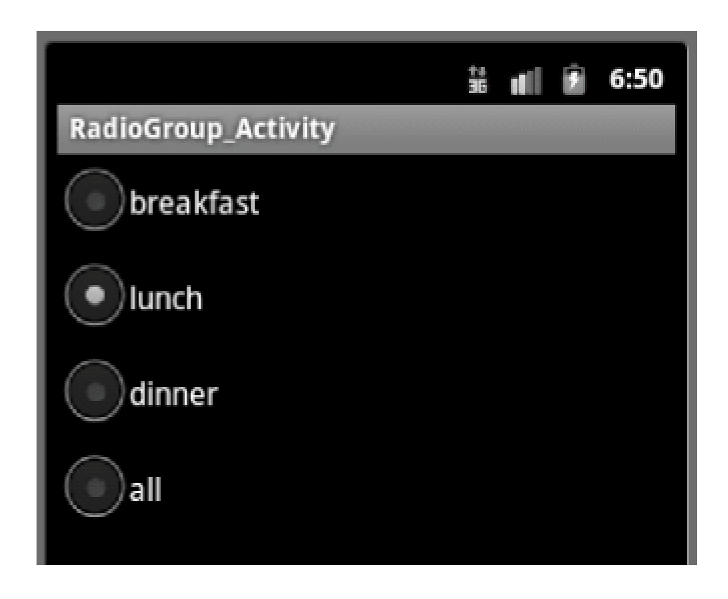


图 4-15　一组 RadioButton 控件

【案例 4.9】　试设计如图 4-15 所示的布局文件片段。

【说明】　这组 RadioButton 控件是将一日的餐名列出，供选择。如果有多个单选按钮组成一组，需要用一个 RadioGroup 控件把它们编成一组，在这个组内，同一时刻只能有一个按钮处于选中状态。

【代码】　布局文件中的代码片段如下所示。

```
<RadioGroup
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:orientation = "vertical"
        android:checkedButton = "@ + id/lunch"
```

```
6           android:id = "@ + id/menu">
7           < RadioButton
8               android:text = "breakfast"
9               android:id = "@ + id/breakfast" />
10          < RadioButton
11              android:text = "lunch"
12              android:id = "@id/lunch" />
13          < RadioButton
14              android:text = "dinner"
15              android:id = "@ + id/dinner" />
16          < RadioButton
17              android:text = "all"
18              android:id = "@ + id/all" />
19  </RadioGroup >
```

(1) 第 1～19 行声明了一个 RadioGroup 控件,其内定义了 4 个 RadioButton 按钮。

(2) 第 4 行设置这组单选按钮是纵向排列的。

(3) 第 5 行设置这个 RadioGroup 组中 ID 为"lunch"的单选按钮为选中状态。

4.3.6 ImageView

1. ImageView(图片)控件

ImageView 控件负责显示图片。除了显示图片之外,还可以对图片执行一些其他的操作,例如设置它的 alpha 值,调整图片的尺寸等。

图片来源有多种途径:可以是资源文件的 ID,也可以是 drawable 目录中的图片对象,还可以是 Content Provider 中的 URI 等。ImageView 控件中常用到的属性和对应方法如表 4-11 所示。

表 4-11 ImageView 中常用的属性和对应方法说明

属 性	方 法	描 述
android:adjustViewBounds	setAdjustViewBounds(boolean)	设置是否需要 ImageView 调整自己的边界来保证显示的图片的长宽比例
android:maxHeight	setMaxHeight(int)	可选项,设置 ImageView 的最大高度
android: maxWidth	setMaxWidth(int)	可选项,设置 ImageView 的最大宽度
android:scaleType	setScaleType(ImageView. ScaleType)	调整或移动图片来适应 ImageView 的尺寸。如:当 ScaleType 取值为 center/CENTER 时按图片的原尺寸居中显示;当 ScaleType 取值为 centerCrop/CENTER_CROP 时按比例扩大图片尺寸并居中显示;当 ScaleType 取值为 fitCenter /FIT_CENTER 时把图片按比例扩大/缩小到 View 的宽度,居中显示;当 ScaleType 取值为 fitXY/FIT_XY 时把图片不按比例扩大/缩小到 View 的大小显示
android:src	setImageResource(int)	设置 ImageView 要显示的图片源

ImageView 控件还有些常用成员方法，如表 4-12 所示。

表 4-12 ImageView 中常用的方法说明

方 法	描 述
setAlpha(int)	设置 ImageView 的透明度
setImageBitmap(Bitmap)	设置 ImageView 所显示的内容为指定的 Bitmap 对象
setImageDrawable (Drawable)	设置 ImageView 所显示的内容为指定的 Drawable 对象
setImageResource (int)	设置 ImageView 所显示的内容为指定 ID 的资源
setImageURI(Uri)	设置 ImageView 所显示的内容为指定 URI
setSelected(boolean)	设置 ImageView 的选中状态

【案例 4.10】 试设计如图 4-16 所示的布局文件片段。

【说明】 这个界面中除了显示了一个水平居中的 ImageView 控件外，在其上方还有一串文本。本界面实际上有两个控件。

【代码】 布局文件中的代码片段如下所示。



图 4-16 一个 ImageView 控件

```
1  <TextView
2          android:layout_width = "wrap_content"
3          android:layout_height = "wrap_content"
4          android:text = "安卓图片是这样的:" />
5
6  <ImageView
7          android:id = "@ + id/imagebutton"
8          android:src = "@drawable/andd"
9          android:layout_width = "wrap_content"
10         android:layout_height = "wrap_content"
11         android:layout_gravity = "center"
12 />
```

(1) 第 1～4 行声明了一个 TextView 控件，显示文本信息。

(2) 第 6～12 行声明了一个 ImageView 控件。其中，第 8 行设置该控件的图片来源于 drawable 目录中的 andd.png 文件；第 11 行设置该图片控件位于屏幕的横向居中位置。

Android 支持常见的静态图片格式，如：PNG、9.PNG、JPG 和 GIF 格式等。注意，Android 还不支持动态的 GIF 格式。这里提到的 9.PNG 格式的图是一种比较特别的图片格式，下面专门介绍。

2. 9Patch 图片简介

9Patch 图片是一种特殊的 PNG 图片格式，以“.9.png”结尾。9Patch 图片可以实现部分拉伸，这种图片与通常的只以“.png”结尾的图片不一样，如果直接拉伸普通的 PNG 图会有失真现象出现，所以它通常被用作背景图使用。

9Patch 图片和普通图片的区别是四周多了一个边框，如图 4-17 所示。

1) 9Patch 图片的伸缩规则

9Patch 图片的伸缩规则主要由左、上侧，右、下侧的黑线确定。一般情况左侧黑线比右侧的短，上侧黑线比下侧的短。右侧和下侧的黑线可以省略。

如图 4-18 所示，以左侧和上侧的黑线两端为标准，可画出 4 条直线，将原图切割成 9 个区域（4 个角，4 条边缘和一个中心区域），各个区域有不同的拉伸控制（角落不拉伸，横边横向拉伸，竖边竖向拉伸，中心区域横竖向都拉伸）。这样，使用 9-patch 图片在伸缩时，其圆角及边缘可避免变形。

如图 4-19 所示，右边的黑色线代表内容绘制的垂直区域，下边的黑色线代表内容绘制的水平区域，以右侧和下侧的黑线两端为标准，可画出 4 条直线。9Patch 图片能显示内容的区域就是这 4 条直线所围的中间区域，其中包含经过伸缩的变形区域。

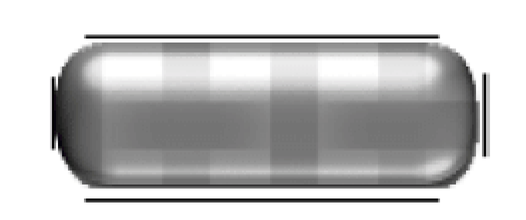


图 4-17 一个作为按钮背景的 9Patch 图片

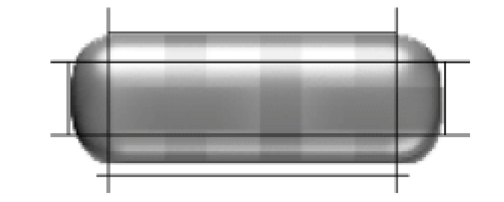


图 4-18 确定伸缩区域

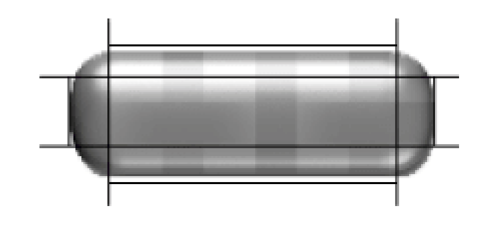


图 4-19 确定显示内容区域

2）9Patch 图片与一般 PNG 图片的比较

通过下面的一组对照图示，如图 4-20 与图 4-21 所示，可以清楚地比较出图. 9. png 与图. png 的区别。

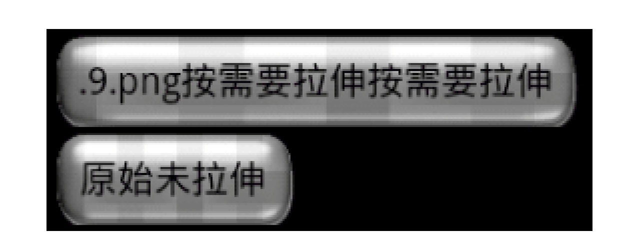


图 4-20 9Patch 图片经过拉伸后与原始图片的比较

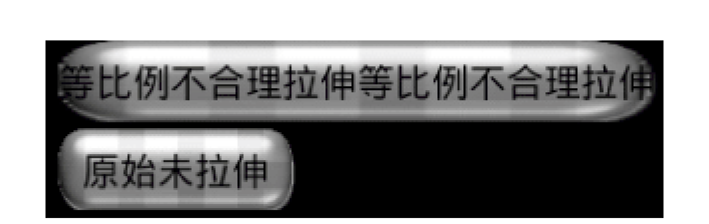


图 4-21 一般的 png 图片经过拉伸后与原始图片的比较

Android 提供了一个制作 9Patch 图片的工具，即在 SDK 的安装文件夹的 tools 文件夹下，可以找到 draw9patch. bat 文件。运行该文件即可打开一个 9Patch 图制作工具，利用它可以将一个普通的图片加工成一个 9Patch 图片。

4.3.7 AnalogClock 和 DigitalClock

在 Android 中显示时钟信息的有 AnalogClock 和 DigitalClock 两个控件。不同的是，AnalogClock 以模拟时钟指针的形式显示时间，且只有时针和分针，而 DigitalClock 以数字形式显示时间，可以精确到秒。它们都位于 android. widget 包下。

【案例 4. 11】 试设计如图 4-22 所示的布局文件片段。

【说明】 这个界面使用了两种控件来显示时间信息，一种是模拟时钟的，一种是以数字形式的，它们都水平居中于屏幕。



图 4-22 时钟显示效果

【代码】 布局文件中的代码片段如下所示。

```
<AnalogClock
    android:id = "@ + id/analog"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_gravity = "center_horizontal"
/>            <! -- 声明一个 AnalogClock 控件 -->
```

```
7  <DigitalClock
8      android:id = "@ + id/digital"
9      android:layout_width = "wrap_content"
10     android:layout_height = "wrap_content"
11     android:layout_gravity = "center_horizontal"
12 />          <! -- 声明一个 DigitalClock 控件 -->
```

（1）第 1～6 行声明了一个 AnalogClock 控件，并水平居中。

（2）第 7～12 行声明了一个 DigitalClock 控件，并水平居中。

4.3.8　DatePicker 和 TimePicker

日期和时间是任何手机都会有的基本功能。Android 也不例外。Android 平台用 DatePicker 控件来显示日期，用 TimePicker 控件来显示时间，并且显示的界面非常精致漂亮。

1. 日期控件

DatePicker 用于向用户提供包含年、月、日的日期数据，并允许用户对其进行修改。如果用户对日期进行了修改，需要使用 onDateChangedListener 监听器来捕获用户修改的日期数据。DatePicker 继承自 FrameLayout 类。其主要的成员方法如表 4-13 所示。

表 4-13　DatePicker 类的主要方法及说明

方　　法	描　　述
getDayOfMonth()	获取日期天数
getMonth()	获取日期月份
getYear ()	获取日期年份
init (int year,int monthOfYear,int dayOfMonth, DatePicker.OnDateChangedListener onDateChangedListener)	初始化 DatePicker 的属性，以 onDateChangedListener 为监听器对象，负责监听日期数据的变化
setEnabled(boolean enabled)	根据传入的参数设置 DatePicker 控件是否可用
updateDate(int year,int monthOfYear,int dayOfMonth)	根据传入的参数更新 DatePicker 控件的各个属性值

【案例 4.12】　试设计如图 4-23 所示的布局文件片段。

【说明】　这就是 Android 的 DatePicker 控件，使用它既可以显示日期，也可以修改日期。因为它继承自 FrameLayout 类，所以它是以屏幕的左上角对齐方式显示的。

【代码】　布局文件中的代码片段如下所示。

```
<DatePicker
    android:id = "@ + id/date_picker"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" />
```



图 4-23　日期选择控件的显示效果

2. 时间控件

TimePicker 用于向用户提供一天中的时间，可以是 24 小时制，也可以为 AM/PM 制，并允许用户对其进行修改。如果要捕获用户修改时间事件，需要添加 onTimeChangedListener

监听器对 TimePicker 进行监听。TimePicker 同样继承自 FrameLayout 类。其常用的成员方法如表 4-14 所示。

表 4-14 TimePicker 类的常用方法及说明

方　法	描　述
getCurrentHour ()	获取 TimePicker 控件的当前小时,返回 Integer 对象
getCurrentMinute ()	获取 TimePicker 控件的当前分钟,返回 Integer 对象
is24HourView ()	判断 TimePicker 控件是否为 24 小时制
setCurrentHour (int currentHour)	设置 TimePicker 控件的当前小时,传入 Integer 对象
setCurrentMinute(int currentMinute)	设置 TimePicker 控件的当前分钟,传入 Integer 对象
setEnabled(boolean enabled)	根据传入的参数设置 TimePicker 控件是否可用
setIs24HourView(boolean is24HourView)	根据传入的参数设置 TimePicker 是否为 24 小时制
setOnTimeChangedListener(TimePicker. OnTimeChangedListener onTimeChangedListener)	为 TimePicker 添加 OnTimeChangedListener 监听器

【案例 4.13】 试设计如图 4-24 所示的布局文件片段。

【代码】 布局文件中的代码片段如下所示。

```
<TimePicker
    android:id = "@ + id/time_picker"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" />
```

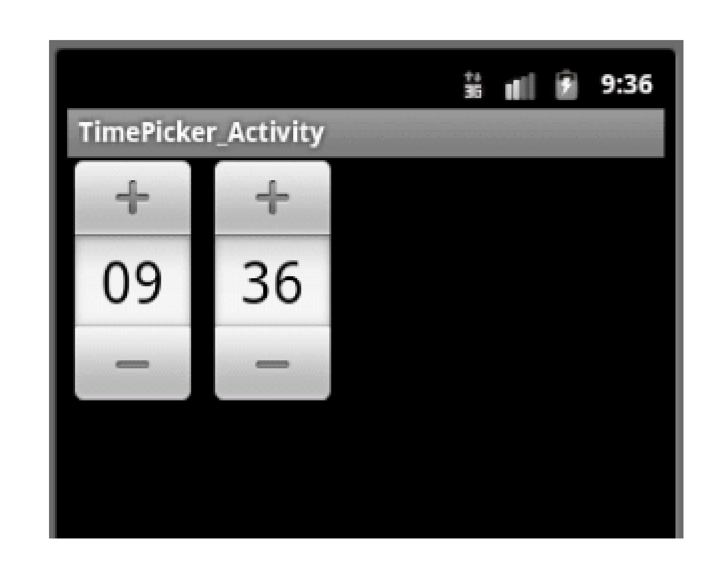


图 4-24 时间选择控件的显示效果

4.4 简单的 UI 设计案例

下面通过一个应用项目的用户界面设计过程来理解本章前三节所介绍的主要内容。

【案例 4.14】 设计一个“掌上微博”的用户登录界面。要求界面友好,使用方便。

【说明】 通常一个用户登录界面需要有这样三个方面的要素:应用项目的名称,用户的账号与密码输入,“登录”和“注册”按钮。

应用项目的名称一般可使用 TextView 控件显示。

用户的账号与密码输入需要用 TextView 控件为输入区作提示,还需要使用编辑控件 EditText,让用户可以进行相关信息输入,还要注意,在输入密码时要将输入的信息隐藏。

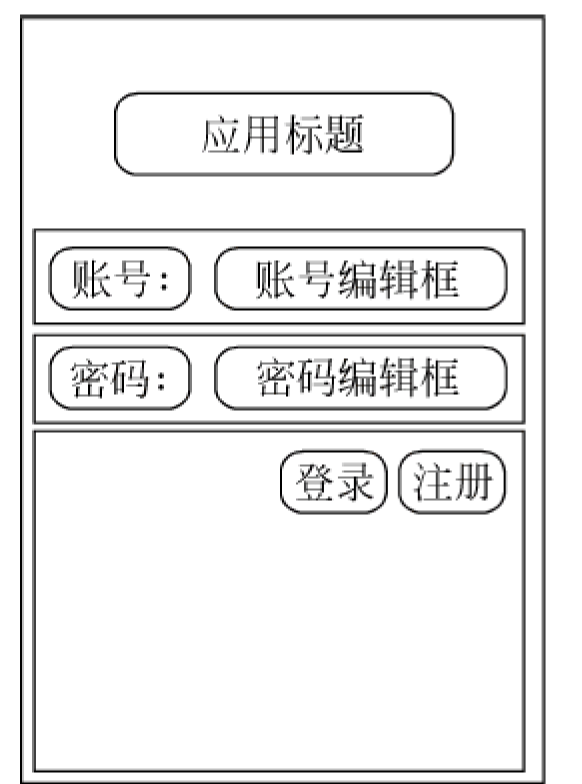


图 4-25 用户登录界面规划

“登录”和“注册”按钮使用 Button 控件就可以了。

为了界面更好看一些,最好配一个背景图作装饰。

为了统一界面的文字表述,需要编写字符串资源文件 strings. xml。

为了统一界面风格,各部分文本的字体、字大小、字颜色、边框等,需要编写样式文件 styles. xml。

【布局设计过程】

(1) 绘出“掌上微博”的用户登录界面草图。在本书中使用的 Android 模拟器屏幕的分辨率是 320×480 像素,按此比例画出一个登录界面的草图如图 4-25 所示。其中矩形表示线性布局

LinearLayout,圆角矩形表示控件。

(2) 从草图中得出以下设计方案。

① 一个外层 LinearLayout。

✍设置 LinearLayout 背景图。

✍为纵向排列布局。

✍一个 TextView 控件显示应用标题,三个 LinearLayout 嵌套。

② 应用标题 TextView 控件。

✍设置宽度为 full_parent,高度为文本高度。

✍设置水平居中对齐方式。

✍设计字体颜色与字体大小,注意与背景图的色彩搭配与大小适中。

③ 账号信息 LinearLayout。

✍为横向排列布局。

✍与上一个对象要有一定间距。

✍一个 TextView 控件显示"账号:",左对齐屏幕,并且设置适当的宽度值。

✍一个 EditText 控件用于输入账号,并且设置适当的宽度值。

④ 密码信息 LinearLayout。

✍与账号信息 LinearLayout 及其内部的控件属性设置基本相同。

✍用于密码输入的 EditText 控件,要设置 android.password 为 true。

⑤ 按钮 LinearLayout。

✍为横向排列布局。

✍与父布局保持一定间距。

✍两个按钮为右对齐方式。

✍设置适当的按钮宽度值。

(3) 在 Eclipse 中创建一个名为 ZSWB_Login1 的 Android 项目。其应用程序名为 ZSWB,包名为 cn.com.sgmsc.ZSWB,Activity 组件名为 Login1Activity。

(4) 将图片资源复制到本项目的 res/drawable-mdpi 目录中。

(5) 编写 res/values 目录下的 strings.xml 文件,代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
        <string name = "hello">Hello World, ZSWB!</string>
        <string name = "app_name">掌上微博</string>
        <string name = "ZSWB">掌上微博</string>
        <string name = "tvUid">账 号: </string>
        <string name = "tvPwd">密 码: </string>
        <string name = "btnLogin">登录</string>
        <string name = "btnReg">注册</string>
</resources>
```

(6) 在 res/values 目录下创建名为 styles.xml 样式文件,并编辑之。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
        <style name = "title">
          <item name = "android:textSize">32sp</item>
```

```
        <item name = "android:textColor"># f37301 </item>
        <item name = "android:textStyle"> bold </item>
    </style>
    <style name = "text">
        <item name = "android:textSize"> 18sp </item>
        <item name = "android:textColor"># f37301 </item>
        <item name = "android:textStyle"> bold </item>
    </style>
    <style name = "button">
        <item name = "android:textSize"> 20sp </item>
        <item name = "android:textColor"># f37301 </item>
        <item name = "android:textStyle"> bold </item>
    </style>
</resources>
```

(7) 重命名 res/layout 目录下的 main.xml 文件为 long.xml，并编辑之。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:gravity = "center_horizontal"
android:background = "@drawable/back"
>                <!-- 声明垂直分布的线性布局 -->

    <TextView
        android:id = "@ + id/TextView0"
        android:text = "@string/ZSWB"
        style = "@style/title"
        android:paddingTop = "20dip"
        android:paddingBottom = "20dip"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:gravity = "center"
        android:password = "false"
        >
    </TextView>

    <LinearLayout
        android:orientation = "horizontal"
        android:paddingTop = "25px"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center_horizontal"
        >
        <TextView
          android:text = "@string/tvUid"
          android:layout_width = "100px"
          style = "@style/text"
          android:layout_height = "wrap_content"
          android:layout_gravity = "center_vertical"
        />
        <EditText
          android:id = "@ + id/etUid"
```

```
40              android:singleLine = "true"
41              android:layout_width = "150px"
42              android:layout_height = "wrap_content"
43            />
44          </LinearLayout>
45
46          <LinearLayout
47            android:orientation = "horizontal"
48            android:layout_width = "wrap_content"
49            android:layout_height = "wrap_content"
50            android:layout_gravity = "center_horizontal"
51          >
52          <TextView
53                android:text = "@string/tvPwd"
54                style = "@style/text"
55                android:layout_width = "100px"
56                android:layout_height = "wrap_content"
57                android:layout_gravity = "center_vertical"
58            />
59          <EditText
60                android:id = "@ + id/etPwd"
61                android:singleLine = "true"
62                android:password = "true"
63                android:layout_width = "150px"
64                android:layout_height = "wrap_content"
65            />
66          </LinearLayout>
67
68          <LinearLayout
69                android:paddingTop = "20dip"
70                android:paddingBottom = "20dip"
71                android:paddingRight = "20dip"
72                android:orientation = "horizontal"
73                android:gravity = "right"
74                android:layout_width = "fill_parent"
75                android:layout_height = "wrap_content"
76            >      <!-- 声明于显示按钮的线性布局 -->
77          <Button
78                android:id = "@ + id/btnLogin"
79                android:text = "@string/btnLogin"
80                style = "@style/button"
81                android:minWidth = "70dip"
82                android:layout_width = "wrap_content"
83                android:layout_height = "wrap_content"
84              />
85          <Button
86                android:id = "@ + id/btnReg"
87                android:text = "@string/btnReg"
88                style = "@style/button"
89                android:minWidth = "70dip"
90                android:layout_width = "wrap_content"
91                android:layout_height = "wrap_content"
92              />
93          </LinearLayout>
94
95  </LinearLayout>
```

① 第 2～9 行声明一个外层的 LinearLayout 布局。其中第 8 行设置了一个背景图，该背

景图片取自于 res/drawable-mdpi 目录中的 back.jpg 文件。

② 第 11～22 行声明一个 TextView 控件用于显示应用标题。其中，第 13 行为控件设置显示的内容，其内容取自字符串资源文件 strings.xml 中名为“ZSWB”的内容；第 14 行设置控件文本显示风格，其文本风格由样式文件 styles.xml 中名为“title”的样式所定义；第 15 行和第 16 行设置该控件与其相邻的上、下控件的间距为 20 像素。

③ 第 24～44 行声明了一个内嵌的 LinearLayout 布局。其中，第 31～37 行声明了一个 TextView 控件，其显示的文本取自于字符串资源文件 strings.xml 中名为“tvUid”的内容，文本显示风格由样式文件 styles.xml 中名为“text”的样式所定义；第 38～43 行声明了一个 EditText 控件，第 40 行设置该 EditText 为单行的编辑框。

④ 第 46～66 行声明了第二个内嵌的 LinearLayout 布局。其中，第 59～65 行声明了一个输入密码的 EditText 控件，第 62 行设置该 EditText 为 password 为 true 的编辑框。

⑤ 第 68～93 行声明了第三个内嵌的 LinearLayout 布局。其中，声明了两个 Button 控件，第 81 行、第 89 行分别设置这两个 Button 的最小宽度为 70 像素，以确保按钮上的文本能显示完整。

由于本例只是在为“掌上微博”的用户登录界面作设计，并没有做任何实质性的代码编写工作，所以就不用编写项目的代码文件和 AndroidManifest.xml 文件了。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 ZSWB_Login1 项目。运行结果如图 4-26 所示。



图 4-26 “掌上微博”用户登录界面

小结

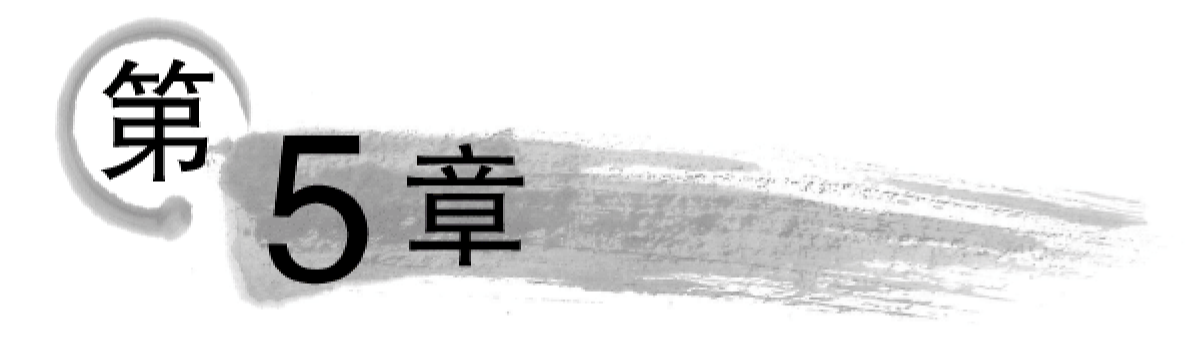
本章简单介绍了 Android 应用程序用户界面的所有控件的基类 View 和所有容器类的基类 ViewGroup。详细介绍了 Android 的 5 种布局类，以及 11 个 Android 的基本控件类。分别通过布局描述文件对这些常用的布局类和基本控件类的属性及其方法进行举例说明。最后通过一个实用程序的用户登录界面设计过程，为读者演示一个应用的用户界面是怎样创建出来的。

是否熟练地掌握了这些布局和基本控件的 XML 描述方法就能够开发出各种各样的用户界面了呢？回答是：否。还有一些屏幕元素如各类视图、选项卡、进度条等控件。是否所有的用户界面都只需要在其布局文件中进行描述，而不需要在代码文件中进行定义呢？答案也是：否。有些控件还必须结合相关的代码才能定义完全。这些知识将在第 5 章中介绍，请继续学习。

练习

1. 设计一个自己微博的登录界面，要求有“账号”和“密码”两个输入编辑框，有“登录”和“注册”两个按钮。

2. 设计一个自己微博的注册界面，要求至少有“账号昵称”、“密码”、“确认密码”等输入编辑框，有“注册”和“返回”两个按钮。



第5章 Android高级控件及事件处理应用

Android 应用程序用户界面的各控件仅能被用户看到，那是“死”的东西，只有它们与用户进行交互，触发了新的动作发生，那么，应用程序就活了。例如，单击一个按钮，进入了另一个界面。或者，单击了下拉框，出现若干选项供选择。又或者，图片或文本开始像走马灯似的动起来等。因此，在开发中，不仅要会在布局文件中描述 UI，还必须在代码文件中做些工作，才能得到一个“活着”的应用程序。

本章将介绍的控件，都需要结合一定的代码编程，才能看到其使用时的完整效果。在介绍它们之前，先来学习 Android 系统的事件处理机制。

5.1 Android 事件处理机制

在 Android 平台上，对事件的处理机制有两种：一种是基于回调机制的事件处理，另一种是基于监听接口的事件处理。

5.1.1 基于回调机制的事件处理

回调机制实质就是将事件的处理绑定在控件上，由图形用户界面控件自己处理事件，回调机制需要自定义 View 来实现。回调不是由该方法的实现方直接调用，而是在特定的事件或条件发生时，由另外的一方通过一个接口来调用，用于对该事件或条件进行响应。

在 Android 平台中，每个 View 都有自己的处理事件的回调方法，开发人员可以通过重写 View 中的这些回调方法来实现需要的响应事件。当某个事件没有被任何一个 View 处理时，便会调用 Activity 中相应的回调方法。

View 类提供了许多公用的捕获用户在界面上触发事件的回调方法，为了捕获和处理事件，必须继承某个类(如 View 类)，并重写这些方法，以便开发者自己定义具体的处理逻辑代码。下面介绍一些常见的回调方法。

1. onKeyDown()方法

几乎所有的 View 都有 onKeyDown()方法，该方法用来捕捉手机键盘被按下的事件。该方法是接口 KeyEvent.Callback 中的抽象方法，所有的 View 全部实现了该接口并重写了该方法。

onKeyDown()方法声明格式：

```
public boolean onKeyDown (int keyCode, KeyEvent event)
```

1）参数说明

（1）参数 keyCode：该参数为 int 类型，表示被按下的键值（即键盘码）。手机键盘中每个键都会有其单独的键盘码，在应用程序中都是通过键盘码才知道用户按下的是哪个键。注意，不同型号的手机中，键值可能不同。

（2）参数 event：该参数为按键事件的对象，其中包含触发事件的详细信息，例如事件的状态、事件的类型、事件发生的时间等。当用户按下按键时，系统会自动将事件封装成 KeyEvent 对象供应用程序使用。

2）返回值

该方法的返回值为一个 boolean 类型的变量。当返回 true 时，表示已经完整地处理了这个事件，并不希望其他的回调方法再次进行处理；而当返回 false 时，表示并没有完全处理完该事件，更希望其他回调方法继续对其进行处理，例如 Activity 中的回调方法。

2. onKeyUp()方法

onKeyUp()方法用来捕捉手机键盘按键抬起的事件。该方法同样是接口 KeyEvent.Callback 中的一个抽象方法，并且所有的 View 同样全部实现了该接口并重写了该方法。

onKeyUp()方法声明格式：

```
public boolean onKeyUp (int keyCode, KeyEvent event)
```

onKeyUp()方法的参数与返回值与 onKeyDown()方法相同，在此不作赘述。

该方法的使用方法与 onKeyDown()基本相同，只是该方法会在按键抬起时被调用。如果用户需要对按键抬起事件进行处理，通过重写该方法即可以实现。

3. onTouchEvent()方法

onTouchEvent()方法用来处理手机屏幕的触摸事件。该方法在 View 类中有定义，并且所有的 View 子类全部重写了该方法。

onTouchEvent()方法声明格式：

```
public boolean onTouchEvent (MotionEvent event)
```

参数说明：

参数 event：为手机屏幕触摸事件封装类的对象，其中封装了该事件的所有信息，例如触摸的位置、触摸的类型以及触摸的时间等。该对象会在用户触摸手机屏幕时被创建。

该方法的返回值机理与键盘响应事件的相同，在此不作赘述。

该方法并不像之前介绍过的方法只处理一种事件，一般情况下以下三种情况的事件全部由 onTouchEvent()方法处理，只是三种情况中的动作值不同。

（1）屏幕被按下：当屏幕被按下时，会自动调用该方法来处理事件，此时 MotionEvent.getAction()的值为 MotionEvent.ACTION_DOWN，如果在应用程序中需要处理屏幕被按下的事件，只需重写该回调方法，然后在方法中进行动作的判断即可。

（2）屏幕被抬起：当触控笔离开屏幕时触发的事件，该事件同样需要 onTouchEvent()方法来捕捉，然后在方法中进行动作判断。当 MotionEvent.getAction()的值为 MotionEvent.ACTION_UP 时，表示是屏幕被抬起的事件。

（3）在屏幕中拖动：该方法还负责处理触控笔在屏幕上滑动的事件，同样是调用

MotionEvent. getAction()方法来判断动作值是否为 MotionEvent. ACTION_MOVE 再进行处理。

下面通过一个简单的案例介绍手机屏幕的触摸事件。

【案例 5.1】 在屏幕区域内触摸滑动,捕捉按下、抬起事件的状态,滑动的坐标,触点压力,触点的大小等信息。

【说明】 在 Java 代码中,有一系列的 get...()方法可用。在此例中需要用到下列方法。

(1) 使用 MotionEvent. getAction()方法来获取屏幕被按下等事件的状态。

(2) 使用 Event. getX()、Event. getY()方法来获取触点坐标值。

(3) 使用 Event. getPressure()方法来获取触屏压力大小。

(4) 使用 Event. getSize()方法来获取触点尺寸。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_Touch 的 Android 项目。其应用程序名为“Touch”,包名为 cn. com. sgmsc. touch,Activity 组件名为 TouchActivity。

(2) 设计布局。编写 res/layout 目录下的 main. xml 文件,代码如下所示。

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3        android:orientation="vertical"
4        android:background="#FF00FF"
5        android:layout_width="fill_parent"
6        android:layout_height="fill_parent">
7        <TextView
8            android:id="@+id/touch_area"
9            android:layout_width="fill_parent"
10           android:layout_height="360dip"
11           android:text="触摸事件测试区"
12           android:textColor="#99FFFF"
13           />
14       <TextView
15           android:id="@+id/event_label"
16           android:layout_width="fill_parent"
17           android:layout_height="wrap_content"
18           android:text="触摸事件: "
19           android:textColor="#FFFFFF"
20           />
21 </LinearLayout>
```

① 第 7～13 行声明了一个 TextView 控件。为了在触摸滑动时将相关的信息显示在屏幕下方,所以在第 10 行设置 TextView 的高为 360dip。

② 第 14～20 行声明另一个 TextView 控件。该控件的资源 id 为“event_label”。

(3) 开发逻辑代码:打开 src/cn. com. sgmsc. touch 包下的 TouchActivity. java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.touch;

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.widget.TextView;
```

```
7
8  public class TouchActivity extends Activity {
9
10     private TextView eventlable;
11
12     @Override
13     public void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.main);
16         eventlable = (TextView) findViewById(R.id.event_label);
17     }
18
19     @Override
20     public boolean onTouchEvent(MotionEvent event) {
21         int action = event.getAction();
22         switch (action) {
23         //当按下的时候
24         case (MotionEvent.ACTION_DOWN):
25             Display("ACTION_DOWN", event);
26             break;
27         //当抬起的时候
28         case (MotionEvent.ACTION_UP):
29             Display("ACTION_UP", event);
30             break;
31         //当触摸的时候
32         case (MotionEvent.ACTION_MOVE):
33             Display("ACTION_MOVE", event);
34         }
35         return super.onTouchEvent(event);
36     }
37
38     public void Display(String eventType, MotionEvent event) {
39         //获取触点相对坐标的信息
40         int x = (int) event.getX();
41         int y = (int) event.getY();
42         //获取触屏压力大小
43         float pressure = event.getPressure();
44         //获取触点尺寸
45         float size = event.getSize();
46         //变量 msg 存放显示信息
47         String msg = "";
48         msg += "事件类型: " + eventType + "\n";
49         msg += "坐标(x , y): " + String.valueOf(x) + " , " + String.valueOf(y) + "\n";
50         msg += "触点压力: " + String.valueOf(pressure) + "\n";
51         msg += "触点尺寸: " + String.valueOf(size) + "\n";
52         eventlable.setText(msg);
53     }
54
55 }
```

① 第 5 行引入 android.view.MotionEvent 类，因为在代码中将使用到触摸滑动类的对象。

② 第 6 行引入 android.widget.TextView 类，因为在代码中要给一个 TextView 类对象赋值。

③ 第 10 行声明一个 TextView 类对象，名为“eventlabel”。

④ 第 12～17 行重写 onCreate()方法。在第 16 行为 eventlabel 对象实例化，即将资源中 ID 号为 event_label 的 TextView 对象赋予变量 eventlabel 中。

⑤ 第 19～36 行重写 onTouchEvent()方法。在该方法中，event 参数是一个 MotionEvent 对象。第 21 行，通过 getAction()方法来获取事件的状态，并将返回结果赋予整型变量 action 中。

⑥ 第 22～34 行是一组 switch-case 语句，根据 action 中的不同值，将不同的参数传入自定义的 Display()方法中。例如，当 action 值为“MotionEvent. ACTION_DOWN”时，调用方法 Display("ACTION_DOWN", event)；当 action 值为“MotionEvent. ACTION_UP”时，调用方法 Display("ACTION_UP", event)；当 action 值为“MotionEvent. ACTION_MOVE”时，调用方法 Display("ACTION_MOVE", event)。

⑦ 第 38～53 行定义了 Display()方法。该方法通过 onTouchEvent()方法调用，在该方法中获取触屏事件的状态、触点坐标、触点尺寸等信息，并且显示在 eventlabel 对象中。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_Touch 项目。运行结果如图 5-1 所示。

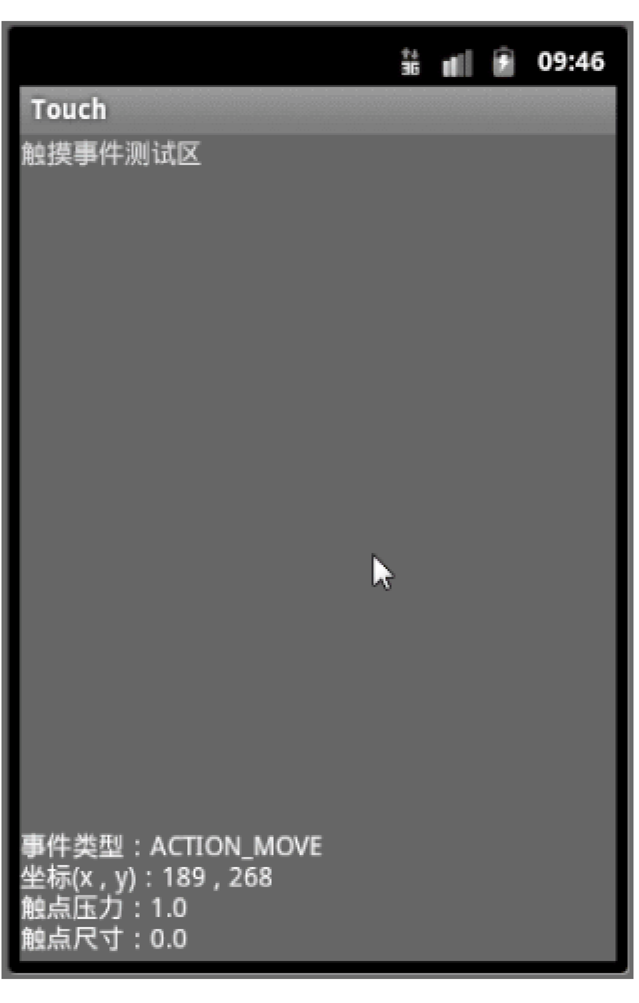
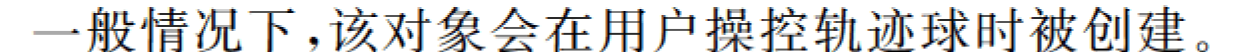


图 5-1　在手机屏幕内触摸滑动及相关信息

4. onTrackballEvent()方法

onTrackballEvent()方法用来处理手机中的轨迹球事件。所有的 View 同样全部实现了该方法。

onTrackballEvent()方法声明格式：



```
public boolean onTrackballEvent (MotionEvent event)
```

参数说明：

参数 event：为手机轨迹球事件封装类的对象，其中封装了触发事件的详细信息，同样包括事件的类型、触发时间等。一般情况下，该对象会在用户操控轨迹球时被创建。

该方法的返回值与前面介绍的各个回调方法的返回值机制完全相同，在此不作赘述。

该方法的使用方法与前面介绍过的各个回调方法基本相同，可以在 Activity 中重写该方法，也可以在各个 View 的实现类中重写。

在手机中使用轨迹球，可以使用户操作达到更好的效果。因为使用轨迹球有如下特点。

(1) 某些型号的手机设计出的轨迹球会比只有手机键盘时更美观。

(2) 轨迹球使用更为简单。

(3) 使用轨迹球会比键盘更为细化，因为滚动轨迹球时，后台的表示状态的数值会变化得更细微、更精准。

如果想在 Android 模拟器中实现轨迹球操作，可以通过 F6 键打开模拟器的轨迹球，然后便可以通过鼠标的移动来模拟轨迹球事件了。

5. onFocusChanged()方法

onFocusChanged()方法用来处理焦点改变的事件。前面介绍的各个方法都可以在 View

及 Activity 中重写，但 onFocusChanged()只能在 View 中重写。当某个控件重写了该方法后，当焦点发生变化时，会自动调用该方法来处理焦点改变的事件。

onFocusChanged()方法声明格式：

```
protected void onFocusChanged (boolean gainFocus, int direction, Rect previouslyFocusedRect)
```

参数说明：

(1) 参数 gainFocus：表示触发该事件的 View 是否获得了焦点，当该控件获得焦点时，gainFocus 等于 true，否则等于 false。

(2) 参数 direction：表示焦点移动的方向，用数值表示，有兴趣的读者可以重写 View 中的该方法打印该参数进行观察。

(3) 参数 previouslyFocusedRect：表示在触发事件的 View 的坐标系中，前一个获得焦点的矩形区域，即表示焦点是从哪里来的。如果不可用则为 null。

该方法没有返回值。

在图形用户界面中，焦点描述了按键事件(或者是屏幕事件)的承受者，每次按键事件都发生在拥有焦点的 View 上。在应用程序中，可以对焦点进行控制，例如从一个 View 移动到另一个 View。下面列出一些与焦点有关的常用方法，如表 5-1 所示。

表 5-1 与焦点有关的常用方法及说明

方　法	描　述
setFocusable (boolean)	设置 View 是否可以拥有焦点
isFocusable ()	监测此 View 是否可以拥有焦点
setNextFocusDownId (int)	设置 View 的焦点向下移动后获得焦点 View 的 ID
hasFocus ()	返回了 View 的父控件是否获得了焦点
requestFocus ()	尝试让此 View 获得焦点
isFocusableTouchMode ()	在触摸模式下，设置 View 控件是否可以拥有焦点。默认情况下是不能的

5.1.2 基于监听接口的事件处理

在 Android 系统中引用 Java 中的事件监听处理机制，它包括事件、事件源和事件监听器三个方面。事件可以是鼠标事件、键盘事件、触摸事件或鼠标移动事件等；事件源是指产生事件的控件；事件监听器是控件产生事件时响应的接口，根据事件的不同重写不同的事件处理方法来处理事件。例如，一辆轿车上安装了防盗设备，当轿车被外力引起强烈震动时就会报警。这时，震动好比事件，轿车好比事件源，报警器好比事件监听器。

1. Android 的监听事件处理模型

对于一个 Android 应用程序来说，事件处理是必不可少的，用户与应用程序之间的交互便是通过事件处理来完成的。在 Android 的监听事件处理模型中涉及以下内容。

1) 事件源与事件

在应用程序中，各个控件在不同情况下触发的事件不尽相同，因此，产生的事件也可能不同。

2）事件监听器

事件监听器则是用来处理事件的对象，实现了特定的接口，根据事件的不同重写不同的事件处理方法来处理事件。

3）事件源与事件监听器

当用户与应用程序交互时，一定是通过触发某些事件来完成的，让事件来通知程序应该执行哪些操作，此过程主要涉及事件源与事件监听器。将事件源与事件监听器联系到一起，就需要为事件源注册监听，当事件发生时，系统才会自动通知事件监听器来处理相应的事件。

在 Android 中为相应接口设置监听器对象方法是使用一系列的 set *** Listener()，为指定的 View 对象设置为 *** 事件接口的监听器。例如，为 Button 对象的 OnClick 事件接口设置监听器使用 setOnClickListener()方法，为在触屏区域的某个 View 对象的 OnTouch 事件接口设置监听器使用 setOnTouchListener()方法，等等。

事件处理的过程一般分为以下三步。

（1）为事件源对象添加监听对象。这样当某个事件被触发时，系统才会知道通知谁来处理该事件。

（2）当事件发生时，系统会将事件封装成相应类型的事件对象，并发送给注册到事件源的事件监听器对象。

（3）当监听器对象接收到事件对象之后，系统会调用监听器中相应的事件处理方法来处理事件并给出响应。

2. 监听器接口与回调方法

正如 Java 中的监听处理模型一样，Android 也提供了同样的基于监听接口的事件处理模型。

1）OnClickListener 接口

（1）功能。

该接口处理的是单击事件。单击事件包括：在触控模式下，在某个 View 上按下并抬起的组合动作；而在键盘模式下，某个 View 获得焦点后单击“确定”键或者按下轨迹球事件。

（2）对应的回调方法。

```
public void onClick(View v)
```

说明：

① 需要实现 onClick()方法。

② 参数 v 便为事件发生的事件源。

2）OnLongClickListener 接口

（1）功能。

OnLongClickListener 接口与之前介绍的 OnClickListener 接口原理基本相同，只是该接口为 View 长按事件的捕捉接口，即当长时间按下某个 View 时触发的事件。

（2）对应的回调方法。

```
public boolean onLongClick(View v)
```

说明：

① 需要实现 onLongClick()方法。

② 参数 v 为事件源控件,当长时间按下此控件时才会触发该方法。

③ 返回值:该方法的返回值为一个 boolean 类型的变量,当返回 true 时,表示已经完整地处理了这个事件,并不希望其他的回调方法再次进行处理;当返回 false 时,表示并没有完全处理完该事件,更希望其他方法继续对其进行处理。

3) OnFocusChangeListener 接口

(1) 功能。

OnFocusChangeListener 接口用来处理控件焦点发生改变的事件。如果注册了该接口,当某个控件失去焦点或者获得焦点时都会触发该接口中的回调方法。

(2) 对应的回调方法。

```
public void onFocusChange(View v, Boolean hasFocus)
```

说明:

① 需要实现 onFocusChange()方法。

② 参数 v 便为触发该事件的事件源。

③ 参数 hasFocus 表示 v 的新状态,即 v 是否获得焦点。

4) OnKeyListener 接口

(1) 功能。

OnKeyListener 是对手机键盘进行监听的接口,通过对某个 View 注册该监听,当 View 获得焦点并有键盘事件时,便会触发该接口中的回调方法。

(2) 对应的回调方法。

```
public boolean onKey(View v, int keyCode, KeyEvent event)
```

说明:

① 需要实现 onKey()方法。

② 参数 v 为事件的事件源控件。

③ 参数 keyCode 为手机键盘的键盘码。

④ 参数 event 便为键盘事件封装类的对象,其中包含事件的详细信息,例如发生的事件、事件的类型等。

5) OnTouchListener 接口

(1) 功能。

OnTouchListener 接口是用来处理手机屏幕事件的监听接口,当在 View 的范围内发生触摸按下、抬起或滑动等动作时都会触发该事件。

(2) 对应的回调方法。

```
public boolean onTouch(View v, MotionEvent event)
```

说明:

① 需要实现 onTouch()方法。对应接口的回调方法。这个方法还处理触摸事件的调用,包括在屏幕上按下、释放和移动手势时调用。

② 参数 v 同样为事件源对象。

③ 参数 event 为事件封装类的对象,其中封装了触发事件的详细信息,同样包括事件的类型、触发时间等信息。

6）OnCreateContextMenuListener 接口

（1）功能。

OnCreateContextMenuListener 接口是用来处理上下文菜单显示事件的监听接口。该方法是定义和注册上下文菜单的另一种方式。

（2）对应的回调方法。

```
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo info)
```

说明：

① 需要实现 onCreateContextMenu()方法。

② 参数 menu 为事件的上下文菜单。

③ 参数 v 为事件源 View，当该 View 获得焦点时才可能接收该方法的事件响应。

④ 参数 info 对象中封装了有关上下文菜单额外的信息，这些信息取决于事件源 View。

该方法会在某个 View 中显示上下文菜单时被调用，开发人员可以通过实现该方法来处理上下文菜单显示时的一些操作。其使用方法与前面介绍的各个监听接口没有任何区别。

3. 事件监听器接口的实现方法

其实这涉及一些 Java 基础知识，主要是接口的一些实现方法。对于事件监听器的实现，有以下三种方式。

（1）在构造方法中使用匿名内部类实现事件监听器接口。通常在 Activity 组件的 onCreate 事件中直接定义，直接动作。其代码片段如下。

```
@override
public void onCreate(Bundle savedInstanceState) {
  …
  Button button1 = (Button)findViewById(R.id. myButton1);
  Button button2 = (Button)findViewById(R.id. myButton2);
  …
  //注册事件监听
  button1.setOnClickListener(new View.OnClickListener() {
    //匿名内部类实现事件监听器接口
    @Override
    public void onClick(View v) {
         …;                                    //执行某些操作
    }
  });
  Button2.setOnClickListener(new View.OnClickListener() {
    //实现方法：匿名内部类实现事件监听器接口
    @Override
    public void onClick(View v) {
       …;                                      //执行某些操作
    }
  });

}
```

这种方式每个控件都定义一次，通常比较不方便。

（2）外部类实现事件监听器接口。通常是在 Activity 组件中实现其接口。其代码片段如下。

```
public class TestMedia extends Activity implements View.OnClickListener{
   …
   @override
   public void onCreate(Bundle savedInstanceState) {
      …
      Button btn1 = (Button) findViewById(R.id.myButton1);
      Button btn2 = (Button) findViewById(R.id.myButton2);

      btn1 .setOnClickListener();
      btn2 .setOnClickListener();

    }
    …
    //实现多个按钮的 OnClick()方法
    @override
    public void onClick(View v) {
       switch (v.getId()) {
       case R.id. myButton1:
         …                                    //do something
         break;
       case R.id. myButton2:
         …                                    //do something
         break;
       }
    }
}
```

这种在 Activity 组件中实现其接口,可以让多个外部控件共享一个接口,即多个控件可以用一个 onClick()方法来定义它们的回调操作。这样相对较方便。

(3) 内部类实现事件监听器接口。第三种类似第二种,其代码片段如下。

```
public class TestMedia extends Activity {
   …
   @override
   public void onCreate(Bundle savedInstanceState) {
      …
      Button btn1 = (Button) findViewById(R.id.myButton1);
      Button btn2 = (Button) findViewById(R.id.myButton2);

      btn1 .setOnClickListener(new ClickEvent());
      btn2 .setOnClickListener(new ClickEvent());

   }
   …
   //实现多个按钮的 OnClick()方法
   class ClickEvent implements View.OnClickListener {
       public void onClick(View v) {
          switch (v.getId()) {
          case R.id. myButton1:
            …                                 //do something
            break;
          case R.id. myButton2:
            …                                 //do something
            break;
          }
```

```
            }
        }
}
```

在一个类中定义一个内部类也可以代表解决问题的一个操作。这样做的好处在于如果需要实现多个监听接口,这种方式更清晰。

这些方式和嵌套接口类都是一一对应的,在开发中如果确定其中一种方式来处理互动事件,就需要在 Activity 中实现相应的带有这个方式的接口,然后通过实例的 set...Listener()方法来设置监听器。例如,调用 setOnClickListener()来设置 OnClickListener 作为监听器。以下三个例子展示了如何设置事件的监听。

【案例 5.2】 对本章的案例 5.1 作改进:将触屏滑动事件局限于一个区域内。

【说明】 案例 5.1 是在整个屏幕范围内捕捉触点的状态信息,现在要求在一个区域内,就必须设置一个控件,在这里将其设置为一个 TextView 区域,并对该区域的 OnTouchListener 接口实现监听。

本例使用第一种接口实现方式,即在构造方法中使用匿名内部类实现事件监听器接口。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_TouchArea 的 Android 项目。其应用程序名为"TouchArea",包名为 cn.com.sgmsc.toucharea,Activity 组件名为 TouchAreaActivity。

(2) 设计布局。编写 res/layout 目录下的 main.xml 文件,代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3       android:orientation = "vertical"
4         android:layout_width = "fill_parent"
5         android:layout_height = "fill_parent">
6         < TextView
7              android:id = "@ + id/touch_area"
8              android:layout_width = "fill_parent"
9              android:layout_height = "360dip"
10             android:background = "#FF00FF"
11             android:text = "触摸事件测试区"
12             android:textColor = "#99FFFF"
13             />
14        < TextView
15             android:id = "@ + id/event_label"
16             android:layout_width = "fill_parent"
17             android:layout_height = "wrap_content"
18             android:text = "触摸事件: "
19             android:textColor = "#FFFFFF"
20             />
21 </LinearLayout>
```

本例中的 main.xml 文件与案例 5.1 的 main.xml 没有太大区别,仅是将触屏事件的测试区背景色改到了第一个 Text View 控件中,见第 10 行中的设置。

(3) 开发逻辑代码。打开 src/cn.com.sgmsc.toucharea 包下的 TouchAreaActivity.java 文件,并编辑之。本例中的代码设计思路与案例 5.1 基本一致,改进之处在于将原来对全屏幕触摸捕捉(即没有指定捕捉对象),改为对一个区域内的触屏捕捉,为此设置了触摸区域,并对该区域的触摸事件进行了监听。代码如下所示。

```java
package cn.com.sgmsc.toucharea;

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.TextView;

public class TouchAreaActivity extends Activity {

    private TextView eventlable;
    private TextView TouchView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TouchView = (TextView) findViewById(R.id.touch_area);
        eventlable = (TextView) findViewById(R.id.event_label);

        TouchView.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                int action = event.getAction();
                switch (action) {
                //当按下的时候
                case (MotionEvent.ACTION_DOWN):
                     Display("ACTION_DOWN", event);
                     break;
                //当抬起的时候
                case (MotionEvent.ACTION_UP):
                     Display("ACTION_UP", event);
                     break;
                //当触摸的时候
                case (MotionEvent.ACTION_MOVE):
                     Display("ACTION_MOVE", event);
                }
                return true;
            }
        });
    }

    public void Display(String eventType, MotionEvent event) {
        //触点相对坐标的信息
        int x = (int) event.getX();
        int y = (int) event.getY();
        //表示触屏压力大小
        float pressure = event.getPressure();
        //表示触点尺寸
        float size = event.getSize();

        String msg = "";
        msg += "事件类型: " + eventType + "\n";
        msg += "相对坐标(x , y): " + String.valueOf(x) + "," + String.valueOf(y) + "\n";
        msg += "触点压力: " + String.valueOf(pressure) + "\n";
```

```
56            msg += "触点尺寸: " + String.valueOf(size) + "\n";
57            eventlable.setText(msg);
58        }
59 }
```

① 第18行实例化一个 TextView 类对象，名为“TouchView”，它取自于资源中 ID 为“touch_area”的控件对象。

② 第21～41行为 TouchView 对象添加了一个监听 setOnTouchListener，同时创建了监听类 View.OnTouchListener()，在这个监听类中匿名地实现了事件监听器接口。

③ 第22～39行重写了 OnTouchListener 接口的回调方法 OnTouch()。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_TouchArea 项目。运行结果如图5-2所示。

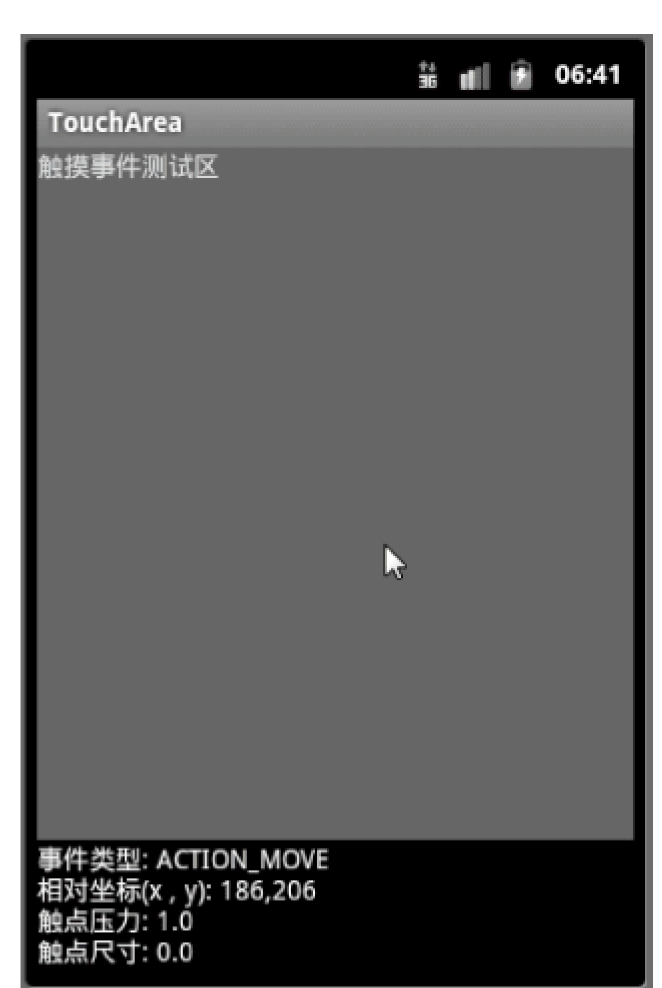


图5-2　获取对有背景色区域内的触摸滑动相关信息

【案例5.3】 将一组单选框中的选择项进行清除。

【说明】 设置一个 Button 对象，当单击这个按钮时，让所有单选按钮的选择状态均为非选状态。为此，需要对该 Button 进行 OnClickListener()监听。

本例使用第二种接口实现方式，即使用外部类实现事件监听器接口。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_RadioGroupClear 的 Android 项目。其应用程序名为“RadioGroupClear”，包名为 cn.com.sgmsc.radiogroupclr，Activity 组件名为 RadioGroupActivity。

(2) 设计布局。编写 res/layout 目录下的布局文件，名为 radio_group.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:orientation = "vertical">
        <RadioGroup
            android:id = "@ + id/menu"
            android:checkedButton = "@ + id/lunch"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:orientation = "vertical"
            >
            <RadioButton
                android:id = "@ + id/breakfast"
                android:text = "breakfast"/>
          <RadioButton
                android:id = "@id/lunch"
                android:text = "lunch"/>
          <RadioButton
                android:id = "@ + id/dinner"
                android:text = "dinner"/>
          <RadioButton
```

```
23                  android:id="@+id/all"
24                  android:text="all"/>
25      </RadioGroup>
26      <Button
27          android:id="@+id/clear"
28          android:layout_width="wrap_content"
29          android:layout_height="wrap_content"
30          android:text="清除"/>
31  </LinearLayout>
```

① 第 6～25 行声明了一个 RadioGroup 控件，其中定义了 4 个 RadioButton，第 8 行设置 ID 为“lunch”的按钮为选中状态。

② 第 26～30 行声明了一个 Button。

(3) 开发逻辑代码。打开 src/cn.com.sgmsc.radiogroupclr 包下的 RadioGroupActivity.java 文件，并编辑之。代码如下所示。

```
1  package cn.com.sgmsc.radiogroupclr;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.view.View;
6  import android.widget.Button;
7  import android.widget.RadioGroup;
8
9  public class RadioGroupActivity extends Activity implements View.OnClickListener {
10
11     private RadioGroup mRadioGroup;
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.radio_group);
17         setTitle("RadioGroup_Activity");
18         mRadioGroup = (RadioGroup) findViewById(R.id.menu);
19         Button clearButton = (Button) findViewById(R.id.clear);
20         clearButton.setOnClickListener(this);
21     }
22
23     @Override
24     public void onClick(View v) {
25         mRadioGroup.clearCheck();
26     }
27 }
```

① 第 3～7 行是一些类的引入。在代码程序中用 View、Button、RadioGroup 三类对象，所以在第 5～7 行要引入 android.view.View、android.widget.Button 和 android.widget.RadioGroup 类。

② 第 9 行声明 RadioGroupActivity 类，并实现 View.OnClickListener 事件监听器接口。

③ 第 13～21 行重写了 onCreate()方法。其中第 18 行 RadioGroup 对象 mRadioGroup 获取实例，第 19 行为 Button 对象 clearButton 获取实例。

④ 第 20 行为 clearButton 对象添加了一个监听 setOnClickListener，这里，“this”指的是本类对象，即 RadioGroupActivity 类。

⑤ 第 23～26 行重写了 View. OnClickListener 接口的回调方法 OnClick()。在此方法中，调用了 RadioGroup 类的 clearCheck()方法，其作用是设置 RadioGroup 组内的单选按钮均为非选中状态。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_RadioGroupClear 项目。运行结果如图 5-3 所示。

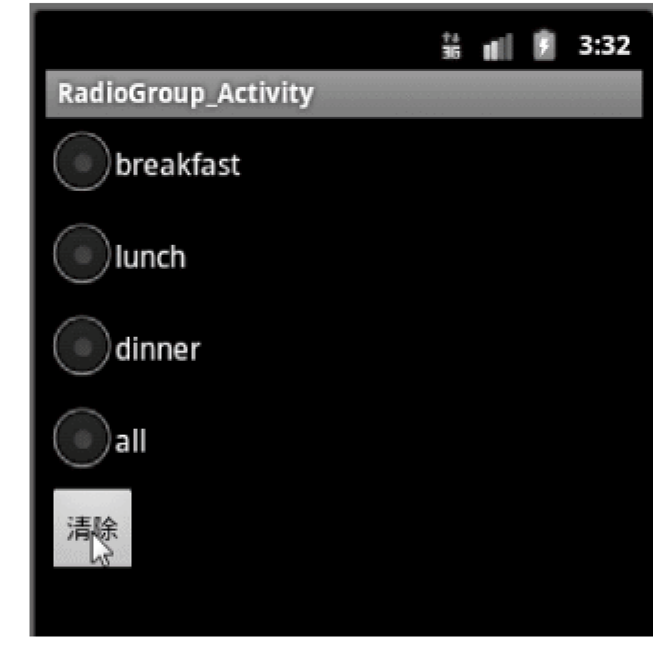


图 5-3　清除单选按钮的选中状态

【案例 5.4】 单击按钮后，将用户输入的信息显示在屏幕的标题栏中。

【说明】 本例中需要有用户的输入操作，因此需要一个编辑框控件 EditText，还需要一个 Button 控件。当用户输入完成后，单击这个按钮，将 EditText 内的信息传送到标题栏中，这个操作将被定义在 Button 的 OnClick ()方法中。

本例使用第三种接口实现方式，即使用内部类实现事件监听器接口。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_Btn 的 Android 项目。其应用程序名为“ButtonClk”，包名为 cn. com. sgmsc. btn，Activity 组件名为 BtnActivity。

(2) 设计布局。编写 res/layout 目录下的布局文件，名为 btn. xml 文件，代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3       android:orientation = "vertical"
4       android:layout_width = "fill_parent"
5       android:layout_height = "fill_parent"
6       >
7
8       <EditText android:id = "@ + id/edit_text"
9            android:layout_width = "fill_parent"
10           android:layout_height = "wrap_content"
11           android:text = "这里可以输入文字" />
12
13      <Button android:id = "@ + id/get_edit_view_button"
14           android:layout_width = "wrap_content"
15           android:layout_height = "wrap_content"
16           android:text = "获取 EditView 的值" />
17 </LinearLayout>
```

一般地，如果在代码中需要用到的控件，都需要向资源中添加它们的 ID。第 8 行、第 13 行为所声明的控件添加 ID 变量。

(3) 开发逻辑代码。打开 src/cn. com. sgmsc. btn 包下的 BtnActivity. java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.btn;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
8
9 public class BtnActivity extends Activity {
10   /** Called when the activity is first created. */
11   @Override
12   public void onCreate(Bundle savedInstanceState) {
13     super.onCreate(savedInstanceState);
14     setTitle("EditText_Activity");
15     setContentView(R.layout.btn);
16     Button get_edit_view_button = (Button) findViewById(R.id.get_edit_view_button);
17     get_edit_view_button.setOnClickListener(new get_edit_view_button_listener());
18   }
19
20   private class get_edit_view_button_listener implements View.OnClickListener {
21       public void onClick(View v) {
22            EditText edit_text = (EditText) findViewById(R.id.edit_text);
23            CharSequence edit_text_value = edit_text.getText();
24           setTitle("输入的值: " + edit_text_value);
25       }
26    };
27 }
```

① 第 16 行，在 onCreate()方法中，获得按钮对象，该按钮名为“get_edit_view_button”。在实际编程中，人们习惯使用控件所执行的操作含义来命名该控件对象。这样做便于程序的维护。

② 第 17 行为 get_edit_view_button 按钮添加一个监听 setOnClickListener，该监听的接口类是 get_edit_view_button_listener。

③ 第 20～26 行，在 BtnActivity 类的内部，定义一个 get_edit_view_button_listener 接口类。在该接口类中重写 onClick()方法。第 23 行使用.getText()方法从 EditText 的对象中取出字符串信息，并赋予字符串变量 edit_text_value 中。第 24 行使用 setTitle()方法将一个字符串信息传送到标题栏中，该信息是“输入的值:”串合并 edit_text_value 中值串的内容。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_Btn 项目。运行结果如图 5-4 所示。



图 5-4　在单击按钮之后获取编辑框中的输入内容

5.2　Android 常用高级控件

在第 4 章中介绍了一个 Spinner 下拉列表控件，当单击这个控件时，会有下拉列表选项。那么这个下拉列表中的选项是怎样定义的呢？又怎样确定用户选择了哪一个选项呢？在 Android 系统中使用了一个适配器的机制来处理这些操作。下面来学习与适配器相关的控件。

5.2.1　与适配器相关的控件

1. AutoCompleteTextView

AutoCompleteTextView 类继承自 EditText 类，位于 android.widget 包下。从外表上

看，AutoCompleteTextView 与 EditText 控件是一样的，只有在用户进行输入中，当输入了与事先为该控件定义的一组字符串集中相关的信息时，才会自动出现下拉选项，供用户选择。例如，事先为该控件定义了一组字符串集“我们，我要，我想，我的，我喜欢，我非常”，当用户在编辑框内输入了一个“我”字，就会在控件下方自动出现下拉选项，将这一系列的“我…”显示出来供用户选择。这有点像在中文输入法中设置了联想输入方式。

AutoCompleteTextView 的某些属性可以在 XML 文件中进行设置，也可以在 Java 代码中通过方法进行设置。下面是 AutoCompleteTextView 控件常用的属性和相应的方法，如表 5-2 所示。

表 5-2　AutoCompleteTextView 常用的属性和相应方法说明

属　　性	方　　法	描　　述
android:completionThreshold	setThreshold (int)	设置用户输入的字符数，当用户输入够该设定的字符数后开始显示下拉列表
android:dropDownHeight	setDropDownHeight (int)	设置下拉列表的高度
android:dropDownWidth	setDropDownWidth (int)	设置下拉列表的宽度
android:popupBackground	setDropDownBackgroundResource(int)	设置下拉列表的背景

从表 5-2 可以看出，这些属性主要用于设置 AutoCompleteTextView 控件下拉时的外形。在下拉列表中的选项内容，需要绑定到数据源上，绑定数据需要用到适配器(Adapter)。

适配器是界面数据绑定的一种理解。它所操纵的数据包括数组、链表、数据库、集合等。适配器就像显示器，把复杂的东西按人可以接受的方式来展现。

在 Android 中有很多的适配器，常用的适配器有 ArrayAdapter、SimpleAdapter、SimpleCursorAdapter，它们都是继承自 BaseAdapter，这些 Adapter 都位于 android. widget 包下。其中以 ArrayAdapter 最为简单，顾名思义，需要把数据放入一个数组中以便显示，一般是展示一行字符。SimpleAdapter 有最好的扩充性，可以自定义出各种效果。SimpleCursorAdapter 可以认为是 SimpleAdapter 对数据库的简单结合，可以方便地把数据库的内容以列表的形式展示出来。

Adapter 对象有两个主要责任：一是用数据填充布局，二是处理用户的选择。下面以 AutoCompleteTextView 控件定义为例，说明 ArrayAdapter 的使用。

【案例 5.5】　AutoCompleteTextView 的使用方法。

【说明】　AutoCompleteTextView 的下拉列表中只是一些字符串，可以使用 String[]数据源来创建一个 ArrayAdapter，为下拉列表进行数据绑定。

创建一个适配器实例，即是为适配器指定显示格式及数据源。实例化 ArrayAdapter 可使用下列方法，其格式为：

```
public ArrayAdapter (Context context, int textViewResourceId, T[] objects)
```

参数 context，为当期的上下文对象。通常使用 this。

参数 textViewResourceId，一个包含 TextView 的布局 XML 文件的 id，用于告诉系统以什么样的布局方式来填充数据。例如，参数值为“android. R. layout. simple_dropdown_item_1line”，这是系统定义好的布局文件，表示在下拉列表中一个数据只显示一行文字串。

参数 objects，是给 ArrayAdapter 提供数据的数组，用来填充下拉列表。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_AutoCompleteTxt 的 Android 项目。其应用程序名为“AutoCompleteTxt”,包名为 cn. com. sgmsc. autocompletetext,Activity 组件名为 AutoCompleteTextViewActivity。

(2) 设计布局。编写 res/layout 目录下的布局文件,名为 autocomplete. xml 文件,代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content">

    <AutoCompleteTextView android:id="@+id/auto_complete"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"/>

</LinearLayout>
```

在布局文件中只声明了一个 AutoCompleteTextView 控件,其资源 id 为“auto_complete”。

(3) 开发逻辑代码。打开 src/cn. com. sgmsc. autocompletetext 包下的 AutoCompleteTextViewActivity. java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.autocompletetext;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class AutoCompleteTextViewActivity extends Activity {
    @Override
     protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.autocomplete);
          setTitle("AutoCompleteTextView_Activity");
          ArrayAdapter<String> adapter  =  new ArrayAdapter<String>(
              this,
              android.R.layout.simple_dropdown_item_1line,
              OUNTRIES);
          AutoCompleteTextView autotextView  =  (AutoCompleteTextView) findViewById(R.id.auto_
          complete);
          autotextView.setAdapter(adapter);
          autotextView.setThreshold(1);
     }
static final String[] COUNTRIES  =  new String[] {
     "China" ,"Russia", "Germany","Ukraine", "Belarus",
     "USA" ,"China1" ,"China12", "Germany1",
     "Russia2", "Belarus1", "USA1"
     };
}
```

① 第3～6行是一些类的引入。在代码程序中用ArrayAdapter和AutoCompleteTextView类对象，所以在第5、6行要引入android.widget.ArrayAdapter和android.widget.AutoCompleteTextView类。

② 第14～17行创建一个适配器并将其实例化。其中第16行使用的是Android系统自带的简单布局，第17行将资源数组OUNTRIES传入。

③ 第18行获取这个控件引用autotextView。

④ 第19行使用setAdapter(adapter)设置适配器。

⑤ 第20行定义用户需要输入的字符数为1，即当用户输入一个字符时就下拉相应的选项列表。

⑥ 第22～26行定义一个名为OUNTRIES的常量数组，作为适配器的资源数组。

【运行结果】 在Eclipse中启动Android模拟器，然后运行Activity_AutoCompleteTxt项目。运行结果如图5-5所示。



图5-5 AutoCompleteTextView输入“c”时的下拉列表

2. Spinner

Spinner(下拉列表)位于android.widget包下。Spinner的外观是一个一行的列表框，右侧有一个下拉按钮，只有当用户单击这个控件时，才会下拉出选项列表供用户选择。

在应用中常常会遇到这样的情况，应用系统已为用户提供了一些选择项，而不需要用户填写内容。这时需要使用Spinner控件。Spinner每次只显示用户选中的元素，当用户再次单击时，会弹出选择列表供用户选择，而选择列表中的元素来自一个适配器，这个选项资源适配器通常在代码中写入。

如果使用ArrayAdapter为Spinner的下拉列表加载数据，有以下两种方式。

(1) 使用Java代码动态地定义下拉列表的数据源。例如向Spinner的下拉列表加载城市名，可使用方法：

```
ArrayAdapter<String>adapter = New ArrayAdapter (this, android.R.layout.simple_spinner_item,
citys)
```

其中，参数“android.R.layout.simple_spinner_item”是系统定义好的布局文件，表示在Spinner未被单击时的显示样式；参数citys是在代码中定义的数组，该数组是预置的一些城市名。

(2) 在res/values目录下，使用XML文件预先定义数据源。例如向Spinner的下拉列表中加载城市名，可使用方法：

```
  ArrayAdapter<CharSequence>adapter = ArrayAdapter.createFromResource(this, R.array.citys,
android.R.layout.simple_spinner_item);
```

其中，参数R.array.citys对应于res/values目录下XML格式的数组资源描述文件，在其中预先定义一组城市名，为Spinner的下拉列表提供数据；参数“android.R.layout.simple_spinner_item”设置在Spinner未被单击时的显示样式。

在Java代码编程中，Spinner控件有一些常用的方法，如表5-3所示。

表 5-3 Spinner 常用的方法说明

方　　法	描　　述
getItemAtPosition(int)	获取在下拉列表中指定位置的数据
getSelectedItem()	获取用户在下拉列表中选定的数据
setDropDownViewResource(int)	设置下拉列表的显示样式，其中的 int 参数是指定布局资源的 ID 号
setPrompt(String)	设置下拉列表框的提示信息
setSelection(int,boolean)	设置 Spinner 在初始化时自动调用一次 OnItemSelectedListener 事件指定的下拉项，如果禁止调用该事件，可使用 setSelection(0,true)

通常，实现一个 Spinner 需要完成以下五个步骤。

(1) 第一步，为下拉列表项定义数据源。

(2) 第二步，实例化一个适配器。

(3) 第三步，为 Spinner 设置下拉列表下拉时的显示样式。

(4) 第四步，将适配器添加到 Spinner 上。

(5) 第五步，为 Spinner 添加监听器，设置各种事件的响应操作。

【案例 5.6】 试设计 Spinner，用于选择所在城市名。

【说明】 在这个 Spinner 中，下拉列表选项只是一些城市名，可以使用 String[]数据源来创建一个 ArrayAdapter，为下拉列表进行数据绑定。

前面已经介绍，为 Spinner 创建 ArrayAdapter 实例有两种方式，因为使用第一种方式创建与案例 5.5 的代码编程差不多，所以本例以使用第二种方式为主来创建 ArrayAdapter 实例(而以注释方式给出第一种方式的编程代码)。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_Spinner 的 Android 项目。其应用程序名为“Spinner”，包名为 cn. com. sgmsc. spinner，Activity 组件名为 SpinnerActivity。

(2) 创建数组资源文件。在 res/values 目录下创建一个名为 arrays. xml 的文件(如果使用第一种方式为 Spinner 的下拉列表加载数据，就不需要创建这个文件)。arrays. xml 文件的代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < resources >
3      < string - array name = "citys">
4          < item >北京</item >
5          < item >上海</item >
6          < item >广州</item >
7          < item >深圳</item >
8          < item >杭州</item >
9          < item >成都</item >
10         < item >大连</item >
11         < item >南京</item >
12     </string - array >
13 </resources >
```

第 3 行定义了这个资源数组的名称为“citys”。注意，在代码中调用此数组资源时，与 XML 文件名无关，而只与“＜string-array name＝"citys"＞”定义的名称“citys”有关。

(3) 设计布局。编写 res/layout 目录下的布局文件，名为 spinner.xml 文件，代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3       android:id = "@ + id/widget28"
4       android:layout_width = "fill_parent"
5       android:layout_height = "fill_parent"
6       android:orientation = "vertical"
7       >
8
9       <TextView
10          android:id = "@ + id/TextView_Show"
11          android:layout_width = "fill_parent"
12          android:layout_height = "wrap_content"
13          android:text = "可以开始选择所在城市了。"
14          android:textSize = "25sp"/>
15
16      <Spinner
17          android:id = "@ + id/spinner_City"
18          android:layout_width = "fill_parent"
19          android:layout_height = "wrap_content"/>
20
21 </LinearLayout>
```

① 第 9～14 行声明了一个 TextView 控件，用于显示从 Spinner 的下拉列表中选择的内容。

② 第 16～19 行声明了一个 Spinner 控件。

(4) 开发逻辑代码。打开 src/cn.com.sgmsc.spinner 包下的 SpinnerActivity.java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.spinner;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;

public class SpinnerActivity extends Activity {

//方式一  声明 citys 字符串数组,为 Spinner 的下拉列表预定义数据:
//方式一      private static final String[] citys = {"北京","上海","广州","深圳","杭州","成都","大连","南京"};
     private TextView text;
     private Spinner spinner;

     @Override
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
```

```
21            setContentView(R.layout.spinner);
22            text = (TextView)findViewById(R.id.TextView_Show);
23            spinner = (Spinner)findViewById(R.id.spinner_City);
24
25            //实例化 ArrayAdapter:
26 //方式一     ArrayAdapter<String> adapter  = new ArrayAdapter<String>(
27 //方式一          this,
28 //方式一          android.R.layout.simple_spinner_item,
29 //方式一          citys);
30            ArrayAdapter<CharSequence> adapter  = ArrayAdapter.createFromResource(
31                 this,
32                 R.array.citys,
33                 android.R.layout.simple_spinner_item);
34
35            //设置 Spinner 的下拉列表显示样式
36            adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
37
38            //将 adapter 添加到 Spinner 中
39            spinner.setAdapter(adapter);
40
41            //设置 Spinner 的一些属性
42            spinner.setPrompt("请选择城市：");
43            spinner.setSelection(0, true);
44

45            //添加 Spinner 事件监听
46            spinner.setOnItemSelectedListener(new Spinner.OnItemSelectedListener(){
47                @Override
48                  public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long
                  arg3) {
49 //方式一                  text.setText("你所在的城市是：" + citysarr[arg2]);
50                    text.setText("你所在的城市是：" + arg0.getItemAtPosition(arg2).toString());
51                      //设置显示当前选择的项
52                      arg0.setVisibility(View.VISIBLE);
53                }
54
55                @Override
56                public void onNothingSelected(AdapterView<?> arg0) {
57                }
58            });
59
60        }
61 }
```

① 第 3～9 行是一些类的引入。在代码程序中使用了 View、AdapterView、ArrayAdapter、Spinner 和 TextView 类对象，所以在第 5～9 行引入了 android.widget.View、android.widget.AdapterView、android.widget.ArrayAdapter、android.widget.Spinner 和 android.widget.TextView 类。

② 第 14 行是使用方式一为 Spinner 的下拉列表加载数据，声明一 String 类型的数组 citys，并为该数组赋初值。另外，我们还可以使用 getStringArray()方法从 XML 数组描述文件中载入数组的值，例如从 arrays.xml 文件中载入数组值方法如下：

```
String citys[] = getResources().getStringArray(R.array.citys);
```

③ 第 18～60 行是重写了 onCreate()方法。

④ 第 26～29 行是使用方式一创建一个名为 adapter 的 ArrayAdapter 实例。这个 ArrayAdapter 的数据来自数组 citys。

⑤ 第 30～33 行使用方式二创建一个名为 adapter 的 ArrayAdapter 实例。这个 ArrayAdapter 的数据来自 arrays.xml 数组描述文件中，注意，使用该方式，ArrayAdapter 的类型必须指定为 CharSequence，不能是 String，否则会出错。

⑥ 第 36 行设置 adapter 将要绑定的 Spinner 对象的下拉列表显示样式。

⑦ 第 39 行为指定的 Spinner 添加适配器 adapte。

⑧ 第 42～43 行为 Spinnerr 设置部分属性，这些属性的设置有助于 UI 的友好性。其中，第 42 行是设置该 Spinner 的下拉列表的提示信息，它可以起到下拉列表的标题作用；第 43 行是保证当项目初始运行时，不让 Spinner 调用 OnItemSelectedListener 事件，因此在 TextView 处不会有 Spinner 的选中项出现，如图 5-6 所示。这两条语句不是必需的，可以省略。

⑨ 第 46～58 行为 Spinner 对象添加一个 OnItemSelectedListener 监听，在其中重定义一些回调方法，这里重写的方法包括 onItemSelected()。

问一下：

onItemSelected()方法中的参数各是什么含义？

onItemSelected()方法的格式为：

```
public void onItemSelected(AdapterView<?> arg0,View arg1, int arg2, long arg3)
```

参数 arg0 指的是适配器视图对象，在这里可以理解为 Spinner 的下拉列表视图。其中，AdapterView 是内容由适配器来决定的视图类，<?>是适配器里内容的类型，可以把“?”理解成你的适配器中的选项数据的类型。

参数 arg1 指的是适配器视图里的被单击的对象。可以理解成下拉列表框中被选中的那一项。

参数 arg2 指在下拉列表选项中被选择项的位置。这个参数类型是 int 型，从 0 开始。

参数 arg3 指被单击选项所在行的行 ID 号。这个参数类型是 long 型。

⑩ 第 49 行使用方式一，取得用户选中的 Spinner 下拉选项值，并将其赋予 TextView 的对象 text 中。

⑪ 第 50 行使用方式二，取得用户选中的 Spinner 下拉选项值，并将其赋予 TextView 的对象 text 中。toString()是将获取的值强制为 String 型。

⑫ 第 52 行，显示用户当前选择项的信息。



图 5-6 初始运行时的显示界面

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_Spinner 项目。初始运行结果如图 5-6 所示，单击了 Spinner 控件后下拉出选项列表，如图 5-7 所示，当在下拉列表中选择了“广州”之后，显示结果如图 5-8 所示。







图 5-7　单击 Spinner 后下拉的列表选项

图 5-8　选择了城市“广州”之后的显示界面

3. ListView

ListView 类位于 android. widget 包下，是 Android 应用开发过程中最常用的控件之一。它是以垂直的可滚动的列表方式显示一组列表项的视图，ListView 里面的每个条目 Item 可以是一个 TextView，也可以是由多个 TextView 和 ImageView 组成的一个组合控件。例如，显示联系人名单、系统设置项等，都会用到 ListView。

实现一个 ListView 控件，主要分为以下 4 个步骤。

(1) 第一步，准备 ListView 要显示的数据，使用一维或多维动态数组保存数据。

(2) 第二步，构建适配器。由于 ListView 的每一个 Item 的组成可能简单，也可能比较复杂，所以根据需要，可选择 ArrayAdapter、SimpleAdapter 或 BaseAdapter 来为 ListView 绑定数据。

(3) 第三步，使用 setAdapter()，把适配器添加到 ListView，并显示出来。

(4) 第四步，为 ListView 添加监听器，设置各种事件(如单击、滚动、单击长按等)的响应操作。

ListView 常用的监听包括：①单击监听，添加单击监听使用 ListView. setOnItemClickListener()；②滚动监听，添加滚动监听使用 ListView. setOnItemSelectedListener()；③长按监听，添加长按监听使用 setOnCreateContextMenuListener()。

1) 使用 ArrayAdapter 适配器

在设计中，使用 ArrayAdapter 适配器为 ListView 绑定数据，可以创建每条目显示一行字符串的 ListView 控件。其具体的实现方法与案例 5. 5 和 5. 6 中创建适配器的方法差不多，如：

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_
item_1, strings);
```

其中，“android. R. layout. simple_list_item_1”是系统定义好的布局文件，表示以一行的文本显示 ListView 的一个 Item 项的样式；“strings”是在代码中定义的数组，也可以是一个列

表数据集合 List,它们为适配器提供数据源,用于 ListView 的显示。

2) 使用 simpleAdapter 适配器

simpleAdapter 的扩展性最好,可以定义各种各样的布局,例如可以放上 ImageView(图片),还可以放上 Button(按钮),CheckBox(复选框)等。使用 simpleAdapter 构造数据一般都是用数组列表 ArrayList,它定义于 java. util. ArrayList 类中,而 ArrayList 一般是通过 HashMap 构成,HashMap 是一组键-值对的集合,HashMap 定义于 java. util. HashMap 类中。

例如,要生成一个 ArrayList 类型的变量 list,并向其中填充两组 HashMap 键-值对 map1、map2,使用代码段如下。

```
1  ArrayList<HashMap<String, String>> list = new  ArrayList<HashMap<String,String>>();
2  //生成两个 HashMap 类型的变量 map1,map2
3  HashMap<String, String>map1 = new HashMap<String, String>();
4  HashMap<String, String>map2 = new HashMap<String, String>();
5  //把数据填充到 map1 和 map2 中
6  map1.put("title", "百度");
7  map1.put("title_ip", " http://www.baidu.com/");
8  map2.put("title", "新浪");
9  map2.put("title_ip", " http://www.sina.com.cn/");
10 //把 map1 和 map2 添加到 list 中
11 list.add(map1);
12 list.add(map2);
```

第 6 行为 map1 的键名为“title”的传值“百度”。第 7 行为 map1 的又一键名为“title_ip”的传值“http://www. baidu. com/”。这个 HashMap 有两个键-值对。

利用 SimpleAdapter 创建适配器实例,其构造方法有 5 个参数,有些参数还比较复杂。例如,在本类中创建一个名为 adapter 的 SimpleAdapter 对象,使用语句如下。

```
SimpleAdapter adapter = new SimpleAdapter(
    this,
    alist,
    R.layout.item,
    new String[]{"img","title","info"},
    new int[]{R.id.img,R.id.title,R.id.info});
```

参数 this 指当前 Activity 的对象。

参数 alist 是一个 ArrayList 类型的列表对象,它向 adapter 中填充数据。

参数 R. layout. item 是一个 XML 布局文件的 ID,ID 所指的布局文件用于设置 ListView 中 Item 的布局。

参数 new String[]{"img","title","info"}是一个 String 类型的数组,该数组中的元素确定了 alist 对象中的列,alist 中有几列对应这个数组中就要有几个元素。

参数 new int[]{R. id. img,R. id. title,R. id. info}是一个 int 类型的数组,该数组中的元素对应着 R. layout. item 所指的布局文件中的控件资源 ID,并且其顺序和个数与上一个参数 String 类型数组中的列名一一对应。例如,String 类型数组的第一个元素是“img”,那么 int 类型数组的第一个元素就是 R. id. img,它是 R. layout. item 布局文件中声明的名为“img”的控件 ID,String 类型数组的第一个元素是“title”,那么 int 类型数组的第一个元素就是 R. id. title,它是 R. layout. item 布局文件中声明的名为“title”的控件 ID,……,如此对应下去。

下面使用一个案例来说明，如何使用 simpleAdapter 适配器为 ListView 控件绑定数据。

【案例 5.7】 使用 SimpleAdapter 适配器为 ListView 绑定数据，列出国内一些著名网站名及网址信息，单击某一条目时，在标题栏显示其网址信息，如图 5-9 和图 5-10 所示。





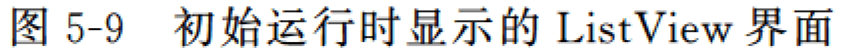


图 5-9 初始运行时显示的 ListView 界面　　图 5-10 单击 ListView 第三项后的界面

【说明】 使用 simpleAdapter 构造数据需要用到 ArrayList，其中的 HashMap 对象对应于 ListView 中的每一条目(Item)。在本例中，ListView 中的每一 Item 包括一个 ImageView 控件和两个分上下行的 TextView 控件。这个布局可以使用一个在 res/layout 目录中的 XML 布局文件来定义。

在本案例中要求每单击 ListView 的一个 Item 项，就要在标题栏显示相关信息，所以需要为该 ListView 对象添加 OnItemClickListener()监听，重写 onItemClick()回调方法。在 onItemClick()方法内执行获取 Item 的信息，并将其显示在标题栏中操作。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_ListViewSimpleAdt 的 Android 项目。其应用程序名为 ListViewSimpleAdt，包名为 cn. com. sgmsc. listviewsimpleadt，Activity 组件名为 ListViewSimpleAdtActivity。

(2) 准备图片资源。将图片资源复制到本项目的 res/drawable-mdpi 目录中，如图 5-11 所示。

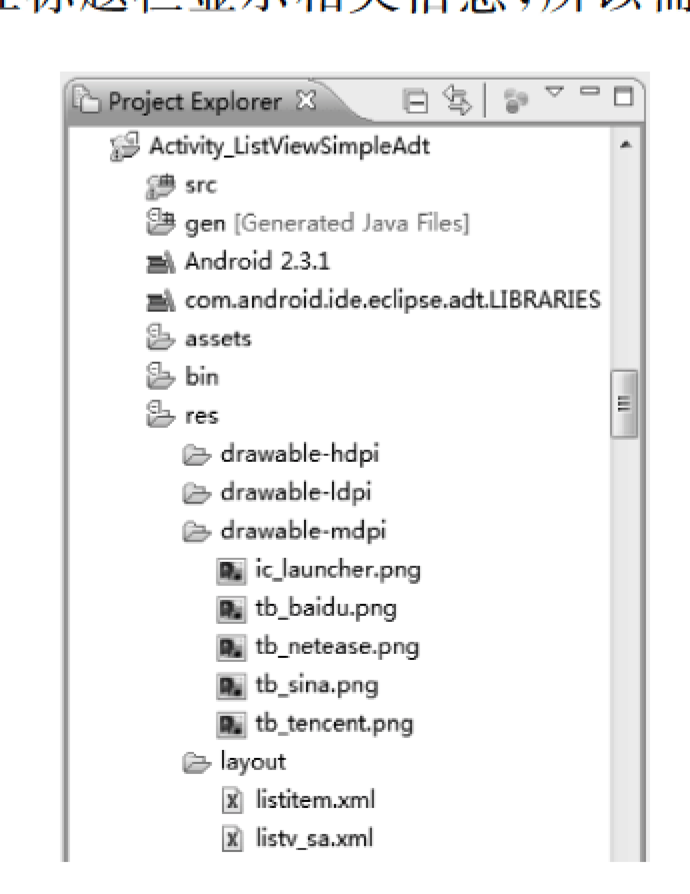


图 5-11 Activity_ListViewSimpleAdt 中的图片资源目录

(3) 设计布局。重命名 res/layout 目录下的 main. xml 文件为 listv_sa. xml，并编辑之。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
     android:id = "@ + id/LinearLayout01"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent"
     >

```

```
8       <ListView
9        android:id = "@ + id/ListView01"
10       android:layout_width = "fill_parent"
11       android:layout_height = "wrap_content"
12       />
13
14 </LinearLayout>
```

第 8～12 行声明了一个 ListView 控件,其 ID 名为 ListView01。

(4) 设计 ListView 的条目布局。在 res/layout 目录下创建一个名为 listitem.xml 的布局文件,该文件为 ListView 的每一个 Item 对象布局,代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3       android:orientation = "horizontal"
4       android:layout_width = "fill_parent"
5       android:layout_height = "fill_parent">
6
7       <ImageView android:id = "@ + id/img"
8            android:layout_width = "49dip"
9            android:layout_height = "50dip"
10           android:layout_margin = "5px"/>
11
12       <LinearLayout android:orientation = "vertical"
13           android:layout_width = "wrap_content"
14           android:layout_height = "wrap_content">
15          <TextView android:id = "@ + id/title"
16               android:layout_width = "wrap_content"
17               android:layout_height = "wrap_content"
18               android:textColor = "#FFFFFFFF"
19               android:textSize = "22px" />
20
21            <TextView android:id = "@ + id/info"
22               android:layout_width = "wrap_content"
23               android:layout_height = "wrap_content"
24               android:textColor = "#FFFFFFFF"
25               android:textSize = "13px" />
26       </LinearLayout>
27 </LinearLayout>
```

① 第 7～10 行声明了一个 ImageView 控件,其 ID 名为 img。为了在 ListView 中每一 Item 的高度一致,并且显示的图片大小也相应一致,在此给出 ImageView 的具体宽度、高度值。

② 第 12～26 行声明一个内嵌的 LinearLayout,用于定义两行文本控件,所以需要设置属性 android:orientation="vertical"。

③ 第 15～19 行声明了一个 TextView 控件,其 ID 名为 title,并设置了文本的大小和文字颜色。

④ 第 21～25 行声明又一个 TextView 控件,其 ID 名为 info,并设置了文本的大小和文字颜色。

（5）开发逻辑代码。打开 src/cn.com.sgmsc.listviewsimpleadt 包下的 ListViewSimpleAdtActivity.java 文件，并编辑之。代码如下所示。

```java
package cn.com.sgmsc.listviewsimpleadt;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;

public class ListViewSimpleAdtActivity extends Activity {

        //private List<String> data = new ArrayList<String>();
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.listv_sa);
            //获得 Layout 里面的 ListView
            ListView list = (ListView) findViewById(R.id.ListView01);

            //生成适配器的 Item 和动态数组对应的元素
            SimpleAdapter listItemAdapter = new SimpleAdapter(
                    this,
                    getData(),
                    R.layout.listitem,
                    new String[]{"img","title","info"},
                    new int[]{R.id.img,R.id.title,R.id.info});

            //添加并且显示
            list.setAdapter(listItemAdapter);

            //添加单击监听
            list.setOnItemClickListener(new OnItemClickListener() {
                   @Override
                   public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
                    Map<String, Object> clkmap = (Map<String, Object>) arg0.getItemAtPosition(arg2);
                      setTitle(clkmap.get("title").toString() + "的网址为: " + clkmap.get("info").toString());
                   }
           });
        }

      //生成多维动态数组，并加入数据
      private List<Map<String, Object>> getData() {
          ArrayList<Map<String, Object>> listitem = new ArrayList<Map<String, Object>>();
           Map<String, Object> map = new HashMap<String, Object>();

           map.put("img", R.drawable.tb_baidu);
```

```
53          map.put("title", "百度");
54          map.put("info", "http://www.baidu.com/");
55          listitem.add(map);
56
57          map = new HashMap<String, Object>();
58          map.put("img", R.drawable.tb_sina);
59          map.put("title", "新浪");
60          map.put("info", "http://www.sina.com.cn/");
61          listitem.add(map);
62
63          map = new HashMap<String, Object>();
64          map.put("img", R.drawable.tb_tencent);
65          map.put("title", "腾讯");
66          map.put("info", "http://www.qq.com/");
67          listitem.add(map);
68
69          map = new HashMap<String, Object>();
70          map.put("img", R.drawable.tb_netease);
71          map.put("title", "网易");
72          map.put("info", "http://www.163.com/");
73          listitem.add(map);
74
75          return listitem;
76      }
77
78 }
```

① 第 3～6 行是一些 Java 类的引入。在代码中使用了 ArrayList、HashMap、List 和 Map 对象，这些对象都定义于 java. util 下的相应包内，所以需要引入 java. util. ArrayList、java. util. HashMap、java. util. List 和 java. util. Map。

② 第 8～14 行引入代码中用到的 Android 中的相关类。

③ 第 18～45 行重写 onCreate()方法。其中，第 24 行从资源中获取 ID 为 ListView01 的 ListView 对象 list；第 27～32 行创建了 SimpleAdater 对象 listItemAdapter，并生成适配器的 Item 和动态数组对应的元素；第 35 行为 list 添加适配器，并显示其内容；第 38～44 行为 list 添加一个单击监听器。

④ 在第 27～32 行创建的 SimpleAdapter 实例中，第二个参数使用的是一个自定义的 getData()方法，该方法返回一个 ArrayList 列表。定义 getData()方法在第 48～76 行。其中第 49 行创建一个 ArrayList 对象 listitem；第 50 行创建一个 HashMap 对象 map，这个 map 内有三组键-值对，它们分别是 img、title 和 info，因为 img 键对应的值是一个图片资源，所以 map 的值不全是 String 类型的，在声明时使用"Map<String, Object>map = new HashMap<String, Object>();"，这里的 String 指定键名的类型，Object 指定键值的类型；第 52～54 行为这个 map 相应的键名传入相应的值内容，第 55 行将 map 对象加入到 listitem 列表中，……，如此下去直到所有的数据都传入完毕；第 75 行将赋有值的 listitem 对象返回。

⑤ 在第 38～44 行设置的监听中，重写了 onItemClick()回调方法。该方法中的 4 个参数与前面讲到的 onItemSelected()方法中的参数含义相似。第 41 行创建一个 Map 对象 clkmap，使用 getItemAtPosition()方法从 arg0(即 ListView 对象)中 arg2 参数所指的那个条目中获取一组 Map 数据给 clkmap；第 42 行从 clkmap 中取出键名为 title 和 info 的值，并将它们显示在标题栏中。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_ListViewSimpleAdt 项目。运行结果满足预期效果。

但是有时候，ListView 不仅是用来显示的，还需要响应 Item 上的操作事件。例如在 Item 中添加了按钮，这时，ListView 需要能够响应其上的按钮单击事件，等等。如果使用 SimpleAdapter，虽然可以在 ListView 中显示出按钮来，但是无法为这个按钮添加监听响应。怎样实现这样的应用呢？此时 BaseAdapter 适配器就派上用场了。

3）使用 BaseAdapter 适配器

BaseAdapter 适配器直接继承接口类 Adapter，BaseAdapter 与其他 Adapter 有些不一样，其他的 Adapter 可以直接在其构造方法中进行数据的设置，但是在 BaseAdapter 中需要实现一个继承自 BaseAdapter 的类，并且继承 BaseAdapter 之后，需要重写以下 4 个方法，这 4 个方法如表 5-4 所示。

表 5-4 继承 BaseAdapter 类需要重写的方法及说明

方 法	描 述
public int getCount()	获取此适配器中所代表的数据集中的条目数
public Object getItem(int position)	获取数据集中与指定位置对应的数据项
public long getItemId(int position)	获取在列表中与指定位置对应的行 ID
public View getView(int position, View convertView, ViewGroup parent)	按指定的布局绘制列表中的每一个 Item 项。其中，position 表示列表中的位置，从 0 开始；convertView 是列表中的 Item 要显示的 View，例如 Spinner、ListView 中每一项要显示的 View；parent 是列表 Item 的父容器，可以理解为如 Spinner、ListView 等控件

ListView 绘制的过程如下：首先，系统在绘制 ListView 之前，将会先调用 getCount()方法来获取 Item 的个数。之后每绘制一个 Item 就会调用一次 getView()方法，在此方法内就可以引用事先定义好的 XML，或在 getView()方法内动态生成显示布局，来确定显示的效果并返回一个 View 对象作为一个 Item 显示出来。如果 getCount()返回值为 0，列表将不显示，如果返回值为 1，则只显示一行。也正是在这个过程中完成了适配器的主要转换功能，把数据和资源以开发者想要的效果显示出来。在定义 ListView 时，getCount()和 getView()方法直接描述了 ListView 的显示效果，而 getItem()和 getItemId()方法将在调用 ListView 的响应方法的时候会被调用到，相比之下，后两者没有前两种方法重要，在案例中有时候省略了其重定义的内容。所以要保证 ListView 的各个方法有效，这两个方法也要重写。例如：如果没有完成 getItemId()方法的功能实现，当调用 ListView 的 getItemIdAtPosition()方法时将会得不到想要的结果，因为该方法就是调用了对应的适配器的 getItemId()方法。下面通过一个案例来学习使用 BaseAdapter 来绑定 ListView 的数据。



图 5-12 初始运行时显示的 ListView 界面

【案例 5.8】 使用 BaseAdapter 适配器为 ListView 绑定数据，并且动态定义显示效果。使得案例 5.7 的 ListView 增加下列特性：每个 Item 添加一个按钮控件，如图 5-12 所示；

当滚动时在标题栏显示其网址信息，如图 5-13 所示；当单击每个条目中的按钮时，在标题栏显示单击的条目 ID 和网址信息，如图 5-14 所示。




图 5-13　执行滚动操作，显示相关信息



图 5-14　单击条目中按钮，显示相关信息

【说明】 在本例中，使用动态生成显示布局的方式，需要在 BaseAdapter 的类对象定义中重写 getView()方法，并在该方法内动态创建布局对象以及其内的控件对象。并且在 BaseAdapter 的类对象定义中需要为每一条目中的按钮添加按钮单击监听方法 OnClickListener()。

为 BaseAdapter 提供的数据源全部放在一个数组描述文件和字符串描述文件中，在代码中使用 getResources(). getIdentifier()方法，通过资源名来获取这些数据的 ID，并加载到适配器中。

为该 ListView 对象添加滚动监听 OnItemSelectedListener()，重写 onItemSelected ()回调方法。在 onItemSelected ()方法内执行获取 Item 的信息，并将其显示在标题栏中操作。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_ListViewBaseAdt 的 Android 项目。其应用程序名为 ListViewBaseAdt，包名为 cn. com. sgmsc. listviewbaseadt，Activity 组件名为 ListViewBaseAdtActivity。

(2) 准备图片资源。将图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 准备字符串资源。编写 res/values 目录下的 strings. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>

    <string name = "hello"> Hello World, ListViewBaseAdtActivity!</string>
    <string name = "app_name"> ListViewBaseAdt </string>

    <string name = "baidu">百度</string>
    <string name = "sina">新浪</string>
    <string name = "tencent">腾讯</string>
    <string name = "netease">网易</string>
    <string name = "baiduurl"> http://www.baidu.com/</string>
    <string name = "sinaurl"> http://www.sina.com.cn/</string>
    <string name = "tencenturl"> http://www.qq.com/</string>
```

```
14        <string name = "neteaseurl">http://www.163.com/</string>
15
16 </resources>
```

第 7～14 行声明了 4 对字符串信息，它们分别是公众网络的名称和对应首页网址。

(4) 创建颜色资源。创建并编写 res/values 目录下的颜色描述文件 colors.xml，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
<color name = "white">#FFFFFFFF</color>
<color name = "red">#FFFD8D8D</color>
<color name = "blue">#FF8D9DFD</color>
</resources>
```

(5) 创建数组资源。创建并编写 res/values 目录下的数组描述文件 arrs.xml，代码如下所示。

```
1 <?xml version = "1.0" encoding = "utf - 8"?>
2 <resources>
3      <string - array name = "images">
4           <item>tb_baidu</item>
5           <item>tb_sina</item>
6           <item>tb_tencent</item>
7           <item>tb_netease</item>
8      </string - array>
9      <string - array name = "titles">
10          <item>baidu</item>
11          <item>sina</item>
12          <item>tencent</item>
13          <item>netease</item>
14     </string - array>
15     <string - array name = "infos">
16          <item>baiduurl</item>
17          <item>sinaurl</item>
18          <item>tencenturl</item>
19          <item>neteaseurl</item>
20     </string - array>
21 </resources>
```

① 第 3～8 行声明了一组图片资源的 id 名数组 images。

② 第 9～14 行声明了一组内容：网站名称的字符串资源的 ID 名数组 titles。

③ 第 15～20 行声明了一组内容：网站地址的字符串资源的 ID 名数组 infos。

(6) 设计布局。重命名 res/layout 目录下的 main.xml 文件为 listv_ba.xml，并编辑之。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
 android:id = "@ + id/LinearLayout01"
 android:layout_width = "fill_parent"
 android:layout_height = "fill_parent"
 >

    <ListView
```

```
        android:id = "@ + id/ListView02"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        />

</LinearLayout>
```

（7）开发逻辑代码。打开 src/cn.com.sgmsc.listviewbaseadt 包下的 ListViewBaseAdtActivity.java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.listviewbaseadt;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemSelectedListener;

public class ListViewBaseAdtActivity extends Activity {

 @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.listv_ba);
        ListView lv = (ListView)this.findViewById(R.id.ListView02);      //初始化 ListView

        BaseAdapter ba = new BaseAdapter(){                        //为 ListView 准备内容适配器

          //所有图片资源名(tb_baidu、tb_sina、tb_tencent、tb_netease)的数组
          String[] images =  getResources().getStringArray(R.array.images);
          //所有标题资源名(baidu、sina、tencent、netease)的数组
          String[] titles = getResources().getStringArray(R.array.titles);
          //所有网址资源名(baiduurl、sinaurl、tencenturl、neteaseurl)的数组
          String[] infos = getResources().getStringArray(R.array.infos);

          public int getCount() {return images.length;} //返回数组的大小,即数组元素个数
          public Object getItem(int arg0) { return null; }
          public long getItemId(int arg0) { return 0; }
          //动态生成每个下拉项对应的 View,每个下拉项 View 由 LinearLayout 中包含一个
          //ImageView,两个内嵌的 LinearLayout 构成,其中一个包含两个 TextView,另一个中包
          //含一个按钮
          public View getView(int arg0, View arg1, ViewGroup arg2) {
            LinearLayout ll0 = new LinearLayout(ListViewBaseAdtActivity.this);
                                                             //初始化 LinearLayout
            ll0.setOrientation(LinearLayout.HORIZONTAL);                   //设置朝向
```

```

ImageView img = new ImageView(ListViewBaseAdtActivity.this);
                                                //初始化 ImageView
img.setImageDrawable(getResources().getDrawable(
  getResources().getIdentifier(images[arg0],"drawable",getPackageName())));
img.setLayoutParams(new Gallery.LayoutParams(49,50));
img.setPadding(5, 5, 5, 5);
ll0.addView(img);                               //添加到 LinearLayout 中

LinearLayout ll1 = new LinearLayout(ListViewBaseAdtActivity.this);
                                                //初始化 LinearLayout
ll1.setOrientation(LinearLayout.VERTICAL);      //设置朝向
ll1.setLayoutParams(new LinearLayout.LayoutParams
    (LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));

TextView tit = new TextView(ListViewBaseAdtActivity.this);  //初始化 TextView
tit.setText(getResources().getText(
    getResources().getIdentifier(titles[arg0],"string",getPackageName())));
                                                //设置内容
tit.setLayoutParams(new LayoutParams(           //设置宽度,高度
    LinearLayout.LayoutParams.WRAP_CONTENT,LinearLayout.LayoutParams.WRAP_CONTENT));
tit.setTextSize(22);                            //设置字体大小
tit.setTextColor(getResources().getColor(R.color.white));    //设置字体颜色
ll1.addView(tit);                               //添加到 LinearLayout 中

TextView inf = new TextView(ListViewBaseAdtActivity.this);  //初始化 TextView
inf.setText(getResources().getText(
    getResources().getIdentifier(infos[arg0],"string",getPackageName())));
                                                //设置内容
inf.setLayoutParams(new LayoutParams(           //设置宽度,高度
    LinearLayout.LayoutParams.WRAP_CONTENT,LinearLayout.LayoutParams.WRAP_CONTENT));
inf.setTextSize(13);                            /设置字体大小
inf.setTextColor(getResources().getColor(R.color.white));    //设置字体颜色
ll1.addView(inf);                               //添加到 LinearLayout 中

ll0.addView(ll1);                               //添加到 LinearLayout 中

LinearLayout ll2 = new LinearLayout(ListViewBaseAdtActivity.this);
                                                //初始化 LinearLayout
ll2.setOrientation(LinearLayout.HORIZONTAL);    //设置朝向
ll2.setLayoutParams(new LinearLayout.LayoutParams
    (LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
ll2.setPadding(3, 0, 3, 0);       //setPadding 参数:(left, top, right, bottom)
ll2.setGravity(Gravity.RIGHT);

Button btn = new Button(ListViewBaseAdtActivity.this);         //初始化 Button
btn.setLayoutParams(new LinearLayout.LayoutParams(50, LayoutParams.WRAP_CONTENT));
btn.setText("显示网址");
btn.setId(arg0);                                //设置 Button 的 ID
//设置按钮的监听器
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      StringBuilder cb = new StringBuilder();  //用 StringBuilder 动态生成信息
      cb.append("单击第" + Integer.toString(v.getId() + 1) + "项,网址为: ");
```

```
94 //                  cb.append("单击第" + String.valueOf(v.getId() + 1) + "项,网址为: ");
95                      cb.append(getResources().getText(
96                              getResources().getIdentifier(infos[v.getId()],"string",
                                getPackageName())));
97                        String clkstr = cb.toString();
98                        setTitle(clkstr);
99                      }
100                 });
101                 ll2.addView(btn);               //将 btn 添加到 ll2 中
102
103                 ll0.addView(ll2);               //将 ll2 添加到 ll0 中
104
105                 return ll0;
106             }
107         };
108
109         lv.setAdapter(ba);                      //为 ListView 设置内容适配器
110
111         lv.setOnItemSelectedListener(           //设置选项选中的监听器
112             new OnItemSelectedListener(){
113               public void onItemSelected(AdapterView<?> arg0, View arg1,int arg2, long arg3) {
114               //重写选项被选中事件的处理方法
115
116                 LinearLayout ll0 = (LinearLayout)arg1;
                                                        //获取当前选中选项对应的 LinearLayout
117                 LinearLayout ll1 = (LinearLayout)ll0.getChildAt(1);
                                                        //获取其中的子 LinearLayout
118
119                 StringBuilder sb = new StringBuilder();      //用 StringBuilder 动态生成信息
120                 TextView tv1 = (TextView)ll1.getChildAt(0);//获取其中的 TextView
121                 sb.append(tv1.getText() + "的网址为: ");
122                 TextView tv2 = (TextView)ll1.getChildAt(1);//获取其中的 TextView
123                 sb.append(tv2.getText());
124                 String stemp = sb.toString();
125                 setTitle(stemp);
126
127                 }
128
129                 public void onNothingSelected(AdapterView<?> arg0) { }
130               }
131         );
132
133     }
134
135 }
```

① 第 3～17 行引入代码中用到的 Android 中的相关类。

② 第 27～107 行创建了一个 BaseAdapter 的类对象 ba,在其中,重定义 ba 中的 getCount(),getItem(),getItemId(),getView()方法,并添加按钮 OnClickListener 监听等。

③ 第 29～34 行定义三个字符串数组,分别用于存放图片、网名、网址资源的 ID 名。

④ 第 36 行重写 getCount()方法,返回数组的大小,用于确定 ListView 的条目数。

⑤ 第 37 行重写 getItem()方法。

⑥ 第 38 行重写 getItemId()方法。

⑦ 第 41～106 行重写 getView()方法。

⑧ 第 42～43 行创建一个 LinearLayout 布局对象 ll0，并设置为横向的。第 45～49 行创建一个 ImageView 控件对象 img，其中第 46 行使用了 4 个方法为 img 获取图片资源。其中使用方法 setImageDrawable(ImageView)将指定的图片赋予 img 对象中并显示出来；使用方法 getResources(). getDrawable(ImageViewId)从指定的资源 ID 中获得图片资源；而 getResources(). getIdentifier(String, String, String)方法是从指定的应用包下，按指定资源名获得 ID 号，如果没有找到指定的资源则返回 0，这里第一个参数资源 ID 名，第二个为资源的类型，第三个为应用所在的包名；使用方法 getPackageName()获得当前所在的包名。

提醒一下：

使用 getResources()方法的注意事项：getResources()方法的功能是获取资源。在使用 getResources()方法时最好在其前面加上限定前缀以明确是谁的资源，即 ListViewBaseAdtActivity. this. getResources()。不过，如果在类的 onCreate()方法内使用 getResources()，可以省略其限定前缀，因为此时所有资源都被默认为 ListViewBaseAdtActivity 类的资源。由于篇幅所限，在本例中都省略了“ListViewBaseAdtActivity. this”。

千万要记住不能在 onCreate 之前使用 getResources()方法。

⑨ 第 48 行设置图片的宽度和高度值，以像素为单位。

⑩ 第 49 行设置图片与四周的间距为 5。

⑪ 第 50 行将 img 对象添加到名为 ll0 的 LinearLayout 布局中。

⑫ 第 52～55 行创建一个内嵌的 LinearLayout 布局 ll1，并设置为纵向，第 54～55 行设置宽度和高度。动态设置子控件的宽度、高度的方法如下，其中第一个参数为宽度，第二个参数为高度：

```
setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_
CONTENT));
```

这条语句其实是子对父而言的，即在父布局下的子控件需要设置这条语句。这时，“new LinearLayout. LayoutParams”说明其父布局为一个 LinearLayout 布局。

⑬ 第 57～64 行和 66～73 行创建了两个 TextView 对象，并把它们添加到 ll1 布局中。

⑭ 第 75 行将 ll1 添加到 ll0 中。

⑮ 第 77～82 行创建新的 LinearLayout 对象 ll2。第 84～86 行在 ll2 内创建一个按钮对象 btn。第 87 行将被单击按钮所在条目位置设置为该按钮的 ID。

⑯ 第 89～100 行为 btn 添加 OnClickListener()监听。在其 OnClick()回调事件中使用 StringBuilder 对象拼接字符串。其中，Integer. toString(v. getId()+1)或 String. valueOf(v. getId()+1)是获取 View 对象的 ID 号，这时的 View 即为 btn。第 95～96 行是 btn 的 ID 所在条目中获取 infos 数组元素所指定的文本内容，并添加到 StringBuilder 对象 cb 中。

问一下：

StringBuilder 是什么类型的数据？

StringBuilder 与 String 类型相似，都用来存放字符串数据。在做字符串拼接时，String 会产生一个新的实例，而 StringBuilder 则不会。所以在大量字符串拼接或频繁对某一字符串进行操作时最好使用 StringBuilder，不要使用 String。

⑰ 第 105 行返回 ll0 布局对象，完成 BaseAdapter 类对象 ba 的创建。

⑱ 第 109 行为 ListView 对象 lv 添加适配器 ba。

⑲ 第 111～132 行为 lv 对象添加滚动监听器。第 113～128 行重写 onItemSelected()方法。在重写中通过(LinearLayout)arg1 获取父布局 ll0；通过(LinearLayout)ll0.getChildAt(1)获取其内的第二个对象，即第一个内嵌的 LinearLayout 对象 ll1；通过(TextView)ll1.getChildAt(0)获取第一个文本对象 tv1；通过 tv1.getText()获取 tv1 中的值；通过(TextView)ll1.getChildAt(1)获取第一个文本对象 tv2，通过 tv2.getText()获取 tv2 中的值；将所得文本字符串拼接在 StringBuilder 对象 sb 中，并显示在标题栏中。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_ListViewBaseAdt 项目。运行结果满足预期效果。

4. GridView

GridView 类位于 android.widget 包下，该视图将其他控件以可滚动的二维网格显示出来，每一个网格项(Item)的显示内容来自于与之相关的 ListAdapter。GridView 是一种较常见的控件，一般用于显示多个图片内容，例如九宫图等。

GridView 类的常用的属性和对应方法如表 5-5 所示。

表 5-5　GridView 常用的属性及对应方法说明

属　性	方　法	描　述
android:columnWidth	setColumnWidth (int)	设置 GridView 的列宽度
android:gravity	setGravity (int)	设置 GridView 中的内容在其内的位置。可选的值有：top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical。可以多选，用"|"分开
android:horizontalSpacing	setHorizontalSpacing (int)	设置两列之间的间距
android:numColumns	setNumColumns (int)	设置 GridView 的列数
android:stretchMode	setStretchMode (int)	设置 GridView 的缩放模式
android:verticalSpacing	setVerticalSpacing(int)	设置两行之间的间距

GridView 的用法很多，下面通过一个完整案例来介绍如何使用 GridView 实现九宫图。

【案例 5.9】 使用 GridView 实现九宫图，每个网格中图片在上方，图片的编号在下方。

【说 明】 GridView 的用法与 ListView 极其类似。在本例中，使用自定义一个 PictureAdapter 类继承 BaseAdapter，再供 GridView 使用。与案例 5.8 不同的是，本例将不采用动态创建布局对象及其内的控件对象，而是采用预先定义好的 XML 布局文件方式。

如果在定义 BaseAdapter 子类时要使用 res/layout 目录下的 XML 布局文件，Android 提供了一个专门的类 LayoutInflater。LayoutInflater 的作用类似于 findViewById()，不同点是 LayoutInflater 是用来找 res/layout 目录下的 XML 布局文件，并且实例化；而 findViewById()则是找具体某一个 XML 下的具体 widget 控件(如：Button、TextView 等)。

为该 GridView 对象添加滚动监听 OnItemClickListener ()，重写 onItemClick ()回调方法，在该方法中执行将单击的网格中的图片编号显示在标题栏中的操作。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_GridView 的 Android 项目。其应用

程序名为 GridView，包名为 cn. com. sgmsc. gridview，Activity 组件名为 GridViewActivity。

（2）准备资源。将图片资源复制到本项目的 res/drawable-mdpi 目录中。这一组图片文件名为 grid_view_01～grid_view_18，有些文件是. gif 格式，有些是. jpg 格式。

（3）设计布局。重命名 res/layout 目录下的 main. xml 文件为 grid_view. xml，并编辑之。代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2
3  <GridView xmlns:android = "http://schemas.android.com/apk/res/android"
4       android:id = "@ + id/gridview"
5       android:layout_width = "fill_parent"
6       android:layout_height = "fill_parent"
7       android:numColumns = "auto_fit"
8       android:verticalSpacing = "10dp"
9       android:horizontalSpacing = "10dp"
10      android:columnWidth = "90dp"
11      android:stretchMode = "columnWidth"
12      android:gravity = "center"
13 />
```

① 第 4 行声明了这个 GridView 控件名为 gridview。

② 第 7 行设置该 GridView 的列数，此时“auto_fit”表示 GridView 的列数设置为自动。

③ 第 11 行设置该 GridView 的缩放模式为按列宽度缩放。

（4）设计 GridView 的单元格布局。在 res/layout 目录下创建一个名为 pic_item. xml 的布局文件，该文件为 GridView 的每一个 Item 对象布局，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
      xmlns:android = "http://schemas.android.com/apk/res/android"
      android:id = "@ + id/root"
      android:orientation = "vertical"
      android:layout_width = "wrap_content"
      android:layout_height = "wrap_content"
      android:layout_marginTop = "5dp"
      >
      <ImageView
          android:id = "@ + id/image"
          android:layout_width = "85dp"
          android:layout_height = "85dp"
          android:layout_gravity = "center"
          android:scaleType = "centerCrop"
          android:padding = "4dp"
          />
      <TextView
          android:id = "@ + id/title"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:layout_gravity = "center"
          />
</LinearLayout>
```

（5）开发逻辑代码。打开 src/cn. com. sgmsc. gridview 包下的 GridViewActivity. java 文

件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.gridview;

import java.util.ArrayList;
import java.util.List;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

public class GridViewActivity extends Activity  {
        private GridView gridView;
        //图片 ID 数组
        private int[] imgs = new int[] {
                R.drawable.grid_view_01, R.drawable.grid_view_02, R.drawable.grid_view_03,
                R.drawable.grid_view_04, R.drawable.grid_view_05, R.drawable.grid_view_06,
                R.drawable.grid_view_07, R.drawable.grid_view_08,R.drawable.grid_view_09,
                R.drawable.grid_view_10, R.drawable.grid_view_11,R.drawable.grid_view_12,
                R.drawable.grid_view_13, R.drawable.grid_view_14,R.drawable.grid_view_15,
                R.drawable.grid_view_16, R.drawable.grid_view_17,R.drawable.grid_view_18
        };
        //图片编号数组
        private String[] tits = new String[] {
                "picture_01", "picture_02", "picture_03",
                "picture_04", "picture_05", "picture_06",
                "picture_07", "picture_08", "picture_09",
                "picture_10", "picture_11", "picture_12",
                "picture_13", "picture_14", "picture_15",
                "picture_16", "picture_17", "picture_18"
        };

        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.griv_view);
            gridView = (GridView) findViewById(R.id.gridview);
            PictureAdapter adapter = new PictureAdapter(tits, imgs, this);
            gridView.setAdapter(adapter);

            gridView.setOnItemClickListener(new OnItemClickListener() {
                    public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
                        if (position<10) {
                            setTitle( "单击的图片是: picture_0" + (position+1));
                        } else  {
                            setTitle( "单击的图片是: picture_" + (position+1));
                      };
```

```
                }
            });
        }
}

//自定义 PictureAdapter 适配器
class PictureAdapter extends BaseAdapter{
        private LayoutInflater inflater;
        private List<Picture> pictures;

        public PictureAdapter(String[] titles, int[] images, Context context) {
            super();
            pictures = new ArrayList<Picture>();
            inflater = LayoutInflater.from(context);
            for (int i = 0; i < images.length; i++)
            {
                Picture pic = new Picture(titles[i], images[i]);
                pictures.add(pic);
            }
        }

        @Override
        public int getCount() {
            if (null != pictures) {
                return pictures.size();
            } else {
                return 0;
            }
        }
        @Override
        public Object getItem(int position) {
            return pictures.get(position);
        }
        @Override
        public long getItemId(int position) { return position; }
        @Override
        public View getView(int position, View convertView, ViewGroup parent)
        {
            ViewHolder viewHolder;
            if (convertView == null)
            {
                convertView = inflater.inflate(R.layout.pic_item, null);
                viewHolder = new ViewHolder();
                viewHolder.title = (TextView) convertView.findViewById(R.id.title);
                viewHolder.image = (ImageView) convertView.findViewById(R.id.image);
                convertView.setTag(viewHolder);
            } else
            {
                viewHolder = (ViewHolder) convertView.getTag();
            }
            viewHolder.title.setText(pictures.get(position).getTitle());
            viewHolder.image.setImageResource(pictures.get(position).getImageId());
            return convertView;
        }
}
```

```
110
111 //定义类 ViewHolder
112 class ViewHolder {
113         public TextView title;
114         public ImageView image;
115 }
116
117 //定义类 Picture
118 class Picture {
119         private String title;
120         private int imageId;
121
122         public Picture() {
123             super();
124         }
125
126         public Picture(String title, int imageId) {
127             super();
128             this.title = title;
129             this.imageId = imageId;
130         }
131
132         public String getTitle() { return title; }
133         public void setTitle(String title) { this.title = title; }
134         public int getImageId() { return imageId; }
135         public void setImageId(int imageId) { this.imageId = imageId; }
136 }
```

① 第 3～17 行引入代码中用到的 Java 和 Android 中的相关类。

② 第 19～58 行定义 GridViewActivity 类。其中,第 21～38 行定义了网格中需要显示的图片 ID 数组和图片编号名数组。第 40～57 行重写 onCreate()方法,第 45 行通过一个自定义的适配器 PictureAdapter 创建适配器对象 adapter,第 48～56 行为 GridView 对象 gridview 添加 OnItemClickListener 监听,并重写 OnItemClick()方法。

③ 第 61～109 行自定义一个适配器 PictureAdapter,它继承自 BaseAdapter。第 65～74 行是定义类 PictureAdapter 的构造方法。第 67 行创建一个 ArrayList 对象,并定义这个 pictures 是由以 Picture(Picture 是一个自定义的类,在代码后面会给出定义)为元素类型的 ArrayList;第 68 行获取 pictures 中每个 Item 的布局;第 69～73 行将指定的 Picture 类型元素一个一个地添加到这个数组列表 pictures 中。在定义子类 PictureAdapter 时还需要重写 getCount()、getItem()、getItemId()、getView()方法。

④ 别忘了,PictureAdapter 类中的具体数据是通过 getView()方法来实现的,所以第 91～108 行是为 PictureAdapter 类实例化的代码段。第 96 行通过 res/layout 目录下的布局文件 pic_item.xml 获得每个网格的显示效果;第 100 行为这个布局添加一个 ViewHolder (ViewHolder 是一个自定义的类,在代码后面会给出定义)类型的数据对象 viewHolder,这里,convertView 中的 setTag(viewHolder)表示给 convertView 添加一个额外的数据,以后可以用 getTag()将这个数据取出来;第 105 行为 viewHolder 对象赋予相应的图片编号字符串并显示;第 106 行为 viewHolder 对象赋予相应的图片并显示。

⑤ 第 112～115 行定义 ViewHolder 类。

⑥ 第 118～136 行定义 Picture 类。其中第 122～124,第 126～130 行是该自定义类的两

个构造方法,第 132～135 行是该自定义类的几个方法,供该类对象调用。

【运行结果】 在 Eclipse 中启动 Android 模拟器,然后运行 Activity_GridView 项目。初始运行结果如图 5-15 所示,单击了 GridView 控件的某个网格后,在标题栏显示相应的信息如图 5-16 所示。



图 5-15 初始运行时的显示界面



图 5-16 单击网格内的图片时显示界面

5. Gallery

Gallery 类位于 android.widget 包下,它是一种较常见的控件,是一种水平滚动的列表,一般用来显示图片等资源,它可以使图片在屏幕上水平地滑来滑去。Gallery 中的图片同样来自适配器。

Gallery 类的常用属性和对应方法如表 5-6 所示。

表 5-6 Gallery 常用的属性及对应方法说明

属 性	方 法	描 述
android:animationDuration	setAnimationDuration (int)	设置动画过渡的时间
android:gravity	setGravity (int)	设置在父控件中的对齐方式
android:unselectedAlpha	setUnselectedAlpha (float)	设置选中图片的透明度
android:spacing	setSpacing(int)	设置图片之间的间距

下面通过一个完整案例来介绍 Gallery 的使用方法。

【案例 5.10】 使用 Gallery 实现一个画廊,该画廊展示 IT 业一些著名人士照片。

【说明】 本例使用 BaseAdapter 为 GridView 提供数据资源。方法与案例 5.8 相似。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_Gallery 的 Android 项目。其应用程序名为 Gallery,包名为 cn.com.sgmsc. gallery,Activity 组件名为 GalleryActivity。

(2) 准备图片。将图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 设计布局。重命名 res/layout 目录下的 main.xml 文件为 gallery.xml,并编辑之。代码如下所示。

```
1 <?xml version = "1.0" encoding = "utf - 8"?>
2 <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3       android:orientation = "vertical"
4       android:layout_width = "fill_parent"
5       android:layout_height = "fill_parent"
6       android:gravity = "center_vertical"
7       >
8      <Gallery
9          android:id = "@ + id/gallery01"
10         android:layout_width = "fill_parent"
11         android:layout_height = "wrap_content"
12         android:spacing = "16dip"
13         android:unselectedAlpha = "1" />
14 </LinearLayout>
```

第 13 行设置被选中项的不透明度为 100%,即最清晰显示照片。

(4) 开发逻辑代码。打开 src/cn.com.sgmsc.gallery 包下的 GalleryActivity.java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.gallery;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.AdapterView.OnItemClickListener;

public class GalleryActivity extends Activity {
 int[] imageIDs = {
            R.drawable.turing,R.drawable.edgar,R.drawable.torvalds,R.drawable.bill,R.drawable.andy
 };

      @Override
      public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.gallery);
          Gallery gl = (Gallery)this.findViewById(R.id.gallery01);

          //创建适配器对象 ba
          BaseAdapter ba = new BaseAdapter(){
            @Override
            public int getCount() {
                  return imageIDs.length;
            }
            @Override
            public Object getItem(int arg0) {
                  return null;
            }
            @Override
            public long getItemId(int arg0) {
                  return 0;
```

```
37              }
38              @Override
39              public View getView(int arg0, View arg1, ViewGroup arg2) {
40                  ImageView iv = new ImageView(GalleryActivity.this);
41                  iv.setImageResource(imageIDs[arg0]);
42                  iv.setScaleType(ImageView.ScaleType.FIT_XY);
43                  iv.setLayoutParams(new Gallery.LayoutParams(220,216));
44                  return iv;
45              }
46          };
47
48          gl.setAdapter(ba);
49
50          //为 gl 添加单击监听
51          gl.setOnItemClickListener(
52              new OnItemClickListener(){
53                  @Override
54                  public void onItemClick(AdapterView<?> arg0, View arg1,
55                          int arg2, long arg3) {
56                      Gallery gly = (Gallery)findViewById(R.id.gallery01);
                                                          //创建一个 Gallery 对象 gly
57                      gly.setSelection(arg2);               //为 gly 设置选中项
58                  }
59              }
60          );
61      }
62 }
```

① 第 14～15 行定义图片资源的 ID 数组 imageIDs。

② 第 25～46 行创建一个 BaseAdapter 对象 ba，并实例化。其中重写 getCount()、getItem()、getItemId()、getView()方法。第 40～43 行是为这个 ba 中的每一项进行赋值的代码段，第 42 行设置图片要横、纵向充满 ImageView 控件对象 iv 的宽、高，第 43 行 iv 的宽与高分别为 220 与 216 像素值。

③ 第 51～60 行为 Gallery 对象 gl 添加 OnItemClickListener 监听。在 onItemClick()回调方法中获取选中的 Gallery 的图片元素。

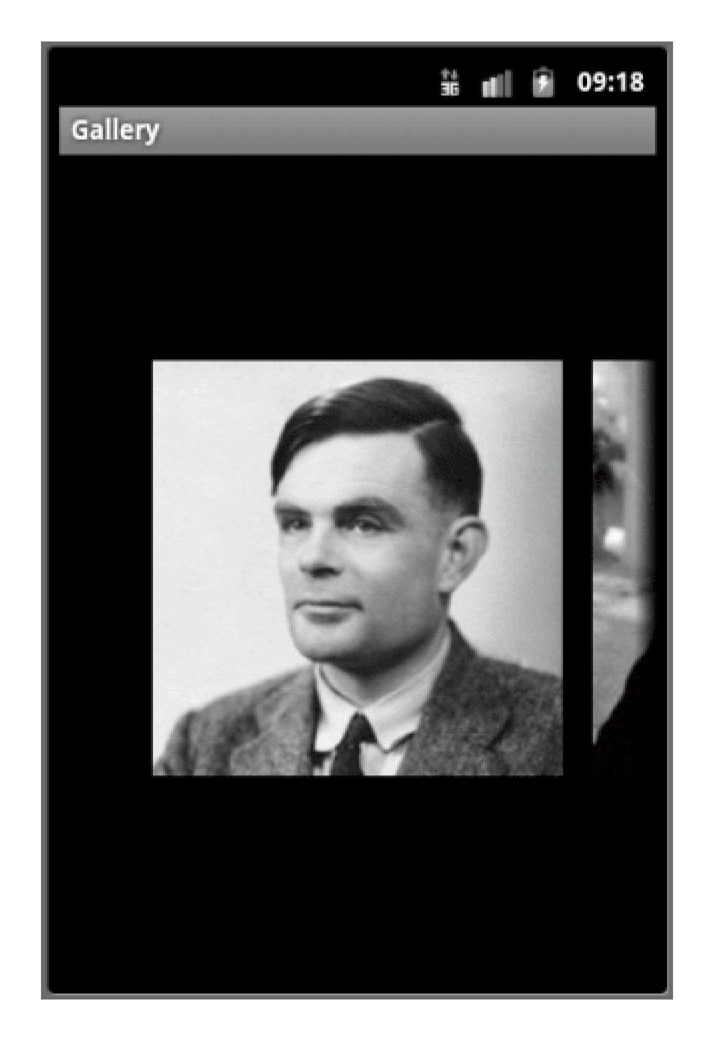


图 5-17　IT 名人画廊显示界面

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_Gallery 项目。运行结果如图 5-17 所示。

在开发中，当需要绑定一些数据展现到屏幕上时，就需要使用 AdapterView 来为控件提供数据源。AdapterView 是 ViewGroup 的子类，它决定了通过 Adapter 来绑定特殊的数据类型并以它定义的布局来展现视图。AutoCompleteTextView、Spinner、ListView、GridView 和 Gallery 都是 AdapterView 的子类。

5.2.2　其他与视图相关的控件

并不是所有展现在屏幕上的一组数据都必须使用 Adapter。有时候，需要显示的项目不多，但一屏显示不下时，可以使用另外一些视图控件来处理。

1. ScrollView

ScrollView 类位于 android. widget 包下，它继承自 FrameLayout，实际上是一个帧布局。在 ScrollView 中可以添加任意一个控件，当其内控件的内容在一屏幕显示不完时，便会自动产生滚动功能，通过纵向滚动的方式以显示被挡住的部分内容。ScrollView 只支持垂直滚动。

注意，ScrollView 中只能加一个控制，不能超过两个。一般情况下，在 ScrollView 布局内添加一个线性布局，然后再将控件添加到这个线性布局中，这样就可以实现在其中添加多个控件了。

2. TabHost

TabHost 类位于 android. widget 包下，它继承自 FrameLayout，是一种帧布局，在其中，它包含多个布局，但同一时刻，根据用户的选择只显示其中一个布局的内容。它是选项卡的封装类，用于创建选项卡窗口。

下面通过一个完整案例来介绍 TabHost 的使用方法。

【案例 5.11】 使用 TabHost 控件分页展示 IT 业一些著名人士信息。

【说明】 在一个帧布局 FrameLayout 内定义多个 LinearLayout 布局，每个 LinearLayout 添加一个名人的信息，其中包括照片和个人简介。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_TabHost 的 Android 项目。其应用程序名为 TabHost，包名为 cn. com. sgmsc. tabhost，Activity 组件名为 TabHostActivity。

(2) 准备图片。将图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 准备字符串资源。编写 res/layout 目录下的 strings. xml 文件，代码如下所示。

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3
4      <string name="hello">Hello World, TabHostActivity!</string>
5      <string name="app_name">TabHost</string>
6
7      <string name="andy">Andy Rubin \n
8          Andy Rubin 是 Google 移动平台资深总监,Android 的创始人之一。</string>
9      <string name="bill">Bill Joy \n
10         是 Java 创造者之一,一位令人崇敬的软件天才,被誉为软件业的爱迪生。</string>
11     <string name="edgar">Edgar F. Codd \n
12         他创造的关系模型是关系数据库的理论基础。被誉为关系数据库之父。</string>
13     <string name="torvalds">Linus Torvalds \n
14         Linus Torvalds,一名芬兰的计算机天才,他是 Linux 系统的创始人。</string>
15     <string name="turing">Turing Alan \n
16         Alan Turing 对早期计算的理论和实践做出了突出的贡献,被誉为 IT 的祖师爷.</string>
17
18 </resources>
```

第 7 行末尾的“\n”表示换行符，作用是其后的文本信息将在下一行显示。

(4) 设计布局。重命名 res/layout 目录下的 main. xml 文件为 tabhost. xml，并编辑之。代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<FrameLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout android:id="@+id/linearLayout01"
        android:layout_width="fill_parent"  android:layout_height="fill_parent"
        android:gravity="center_horizontal"  android:orientation="vertical">
     <ImageView
            android:id="@+id/tab_ImageView01"
            android:scaleType="fitXY"
            android:layout_gravity="center"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/andy"/>
      <TextView
            android:id="@+id/tab_TextView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="24dip"
            android:textColor="#f37301"
            android:text="@string/andy"/>
    </LinearLayout>
    <LinearLayout android:id="@+id/linearLayout02"
        android:layout_width="fill_parent"  android:layout_height="fill_parent"
        android:gravity="center_horizontal"  android:orientation="vertical" >
      <ImageView
            android:id="@+id/tab_ImageView02"
            android:scaleType="fitXY"
            android:layout_gravity="center"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/bill"/>
     <TextView
            android:id="@+id/tab_TextView02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="24dip"
            android:textColor="#f37301"
            android:text="@string/bill"/>
   </LinearLayout>
   <LinearLayout android:id="@+id/linearLayout03"
        android:layout_width="fill_parent"  android:layout_height="fill_parent"
        android:gravity="center_horizontal"  android:orientation="vertical">
      <ImageView
            android:id="@+id/tab_ImageView03"
            android:scaleType="fitXY"
            android:layout_gravity="center"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/torvalds"/>
      <TextView
            android:id="@+id/tab_TextView03"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="24dip"
            android:textColor="#f37301"
```

```
57                 android:text = "@string/torvalds"/>
58         </LinearLayout >
59         < LinearLayout android:id = "@ + id/linearLayout04"
60               android:layout_width = "fill_parent"    android:layout_height = "fill_parent"
61               android:gravity = "center_horizontal"   android:orientation = "vertical">
62            < ImageView
63                 android:id = "@ + id/tab_ImageView04"
64                 android:scaleType = "fitXY"
65                 android:layout_gravity = "center"
66                 android:layout_width = "wrap_content"
67                 android:layout_height = "wrap_content"
68                 android:src = "@drawable/edgar"/>
69            < TextView
70                 android:id = "@ + id/tab_TextView04"
71                 android:layout_width = "wrap_content"
72                 android:layout_height = "wrap_content"
73                 android:textSize = "24dip"
74                 android:textColor = "#f37301"
75                 android:text = "@string/edgar"/>
76         </LinearLayout >
77         < LinearLayout android:id = "@ + id/linearLayout05"
78               android:layout_width = "fill_parent"    android:layout_height = "fill_parent"
79               android:gravity = "center_horizontal"   android:orientation = "vertical">
80            < ImageView
81                 android:id = "@ + id/tab_ImageView05"
82                 android:scaleType = "fitXY"
83                 android:layout_gravity = "center"
84                 android:layout_width = "wrap_content"
85                 android:layout_height = "wrap_content"
86                 android:src = "@drawable/turing"/>
87            < TextView
88                 android:id = "@ + id/tab_TextView05"
89                 android:layout_width = "wrap_content"
90                 android:layout_height = "wrap_content"
91                 android:textSize = "24dip"
92                 android:textColor = "#f37301"
93                 android:text = "@string/turing"/>
94         </LinearLayout >
95     </FrameLayout >
```

第 2～95 行声明了一个 FrameLayout，其中声明了 5 个 LinearLayout。

(5) 开发逻辑代码。打开 src/cn. com. sgmsc. tabhost 包下的 TabHostActivity. java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.tabhost;

import android.app.TabActivity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.widget.TabHost;

public class TabHostActivity extends TabActivity {
     private TabHost myTabhost;
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
```

```
12          myTabhost = this.getTabHost();          //从 TabActivity 上面获取放置 Tab 的 TabHost
13          LayoutInflater.from(this).inflate(R.layout.tabhost, myTabhost.getTabContentView(),
            true);
14          myTabhost.addTab(
15                myTabhost.newTabSpec("选项卡 1")  //新建一个名为"选项卡 1"的 Tab 标签
16                .setIndicator("Andy",getResources().getDrawable(R.drawable.andy))
                                                    //设置 Tab 的标题内容
17                .setContent(R.id.linearLayout01)  //设置此 Tab 页显示的布局控件
18                );
19          myTabhost.addTab(
20                myTabhost.newTabSpec("选项卡 2")
21                .setIndicator("Bill", getResources().getDrawable(R.drawable.bill))
22                .setContent(R.id.linearLayout02)
23                );
24          myTabhost.addTab(
25                myTabhost.newTabSpec("选项卡 3")
26                .setIndicator("Torvalds", getResources().getDrawable(R.drawable.torvalds))
27                .setContent(R.id.linearLayout03)
28                );
29          myTabhost.addTab(
30                myTabhost.newTabSpec("选项卡 4")
31                .setIndicator("Edgar", getResources().getDrawable(R.drawable.edgar))
32                .setContent(R.id.linearLayout04)
33                );
34          myTabhost.addTab(
35                myTabhost.newTabSpec("选项卡 5")
36                .setIndicator("Turing", getResources().getDrawable(R.drawable.turing))
37                .setContent(R.id.linearLayout05)
38                );
39      }
40  }
```

① 第 8 行声明 TabHostActivity 类是继承 TabActivity 类的，所以第 3 行要引入 android.app.TabActivity。

② 第 12 行从 TabActivity 上面获取 TabHost 对象 myTabhost。

③ 第 13 行从布局文件 tabhost.xml 中获取显示模板。

④ 第 14～18 行设置第一个选项卡的内容。其中，第 15 行是在 myTabhost 上添加一个名为"选项卡 1"的选项卡；第 16 行设置这个选项卡的标题内容，标题名为"Andy"，标题画片取自图片资源 R.drawable.andy；第 17 行设置选项卡显示的内容为 tabhost.xml 布局中的名为 LinearLayout01 的线性布局所指定的内容……如此下去，设置了后面 4 个选项卡的内容。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_TabHost 项目。运行结果如图 5-18 所示。



图 5-18 执行选项卡案例的显示界面

3. ImageSwitcher

ImageSwitcher 类位于 android.widget 包下，它是 Android 中控制图片展示效果的一个控件，常常与 Gallery 控件一起使

用，可以实现多张图片进行切换的幻灯片效果。

ImageSwitcher 获取图片的常用方法如下。

(1) setImageURI(Uri uri)，从 URI 地址获取图片设置到 ImageSwitcher 中。

(2) setImageResource(int resId)，从图片资源库 ID 获取图片设置到 ImageSwitcher 中。

(3) setImageDrawable(Drawable drawable)，从 drawable 目录中获取图片设置到 ImageSwitcher 中。

下面通过一个完整案例来介绍 ImageSwitcher 和 Gallery 的使用方法。

【案例 5.12】 使用 ImageSwitcher 和 Gallery 控件制作一个浏览图片的展示窗，这个展示窗分为上下两部分：在屏幕上方展示当前选中的大图片；在屏幕的下方是一组可以滚动的小图片，位于中央位置的小图片即为当前选中的图片。

【说明】 使用 ImageSwitcher 作为大图片的展示控件，使用 Gallery 作为小图片的滚动控件。布局文件可选用相对布局来实现。

在 ImageSwitcher 的类声明时要实现一个 ViewSwitcher. ViewFactory 接口。这个接口里有一个方法 makeView()，在实现时需要定义它。这个方法为 ImageSwitcher 返回一个视图 View 对象。ImageSwitcher 的调用过程是：首先要有一个 Factory 为它提供一个 View，然后 ImageSwitcher 就可以初始化各种资源了。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_ImageSwitch 的 Android 项目。其应用程序名为 ImageSwitch，包名为 cn. com. sgmsc. imageswitch，Activity 组件名为 ImageSwitchActivity。

(2) 准备图片。将图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 设计布局。重命名 res/layout 目录下的 main. xml 文件为 image_switch. xml，并编辑之。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout
        xmlns:android = "http://schemas.android.com/apk/res/android"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">

        <ImageSwitcher
        android:id = "@ + id/switcher"
                android:layout_width = "fill_parent"
                android:layout_height = "fill_parent"
                android:layout_alignParentTop = "true"
                android:layout_alignParentLeft = "true" />

       <Gallery android:id = "@ + id/gallery"
                android:background = "#55000000"
                android:layout_width = "fill_parent"
                android:layout_height = "60dp"
                android:layout_alignParentBottom = "true"
                android:layout_alignParentLeft = "true"
                android:gravity = "center_vertical"
                android:spacing = "16dp"
                android:unselectedAlpha = "1"/>

```

```
24 </RelativeLayout>
```

① 第 2～24 行声明了一个相对布局。

② 第 7～12 行声明了一个 ImageSwitch 控件，与其父控件的左、上方对齐，用于显示选中的大图片。

③ 第 14～22 行声明了一个 Gallery 控件，与其父控件的左、下方对齐，用于显示横向滚动的小图标列表索引。其中，第 15 行设置这个 Gallery 的背景色为半透明的黑色，这样设置，如果大图片的尺寸较大，图片部分与 Gallery 控件有重叠时，半透明的背景色不会遮住大图片的显示。

(4) 开发逻辑代码。打开 src/cn. com. sgmsc. imageswitch 包下的 ImageSwitchActivity. java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.imageswitch;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.ViewSwitcher;
import android.widget.Gallery.LayoutParams;

public class ImageSwitchActivity  extends Activity implements
AdapterView.OnItemSelectedListener, ViewSwitcher.ViewFactory {

    private ImageSwitcher mySwitcher;
    private Integer[] myThmIds = {
             R.drawable.turing_sw, R.drawable.edgar_sw, R.drawable.torvalds_sw,
             R.drawable.bill_sw, R.drawable.andy_sw};
    private Integer[] myImgIds = {
           R.drawable.turing,R.drawable.edgar,R.drawable.torvalds,
           R.drawable.bill,R.drawable.andy};

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);  //设置这个 Activity 没有标题栏
        setContentView(R.layout.image_switch);

        mySwitcher = (ImageSwitcher) findViewById(R.id.switcher);
        mySwitcher.setFactory(this);
        mySwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_in));
        mySwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_out));

```

```
42         Gallery g = (Gallery) findViewById(R.id.gallery);
43         g.setAdapter(new ImageAdapter(this));
44         g.setOnItemSelectedListener(this);
45     }
46
47     public void onItemSelected(AdapterView<?> parent, View v, int position, long id) {
48       mySwitcher.setImageResource(myImgIds[position]);  //为 mSwitcher 设选中的图片资源
49     }
50
51     public void onNothingSelected(AdapterView<?> parent) {
52     }
53
54     public View makeView() {
55         ImageView iv = new ImageView(this);
56         iv.setBackgroundColor(0xFF000000);              //设置背景颜色
57         iv.setScaleType(ImageView.ScaleType.FIT_CENTER);  //
58         iv.setLayoutParams(new ImageSwitcher.LayoutParams(LayoutParams.FILL_PARENT,
59                 LayoutParams.FILL_PARENT));
60         return iv;
61     }
62
63     //自定义 ImageAdapter 适配器
64     public class ImageAdapter extends BaseAdapter {
65         private Context myContext;
66         public ImageAdapter(Context c) { myContext = c; }
67         public int getCount() { return myThmIds.length; }
68         public Object getItem(int position) { return position; }
69         public long getItemId(int position) { return position; }
70         public View getView(int position, View convertView, ViewGroup parent) {
71             ImageView i = new ImageView(myContext);
72             i.setImageResource(myThmIds[position]);
73             i.setAdjustViewBounds(true);
74             i.setLayoutParams(new Gallery.LayoutParams(
75                     LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
76             return i;
77         }
78     }
79
80 }
```

① 第 18～19 行声明 ImageSwitchActivity 类是一个继承接口 AdapterView.OnItemSelectedListener 和 ViewSwitcher.ViewFactory 的类。

② 第 22～24 行声明一个图片资源的 ID 数组 myThmIds，为 Gallery 准备数据，第 25～27 行声明一个图片资源的 ID 数组 myImgIds，为 ImageSwitch 准备数据。

③ 第 32 行设置这个 Activity 的标题栏不可见。注意，如果要设置窗口显示的属性，如标题栏不可见，必须放在“setContentView(…);”语句之前。

④ 第 35～40 行设置 ImageSwitcher 对象 mySwitcher。首先在第 35 行从资源中获取 ImageSwitcher 的实例；然后第 36 行调用 setFactory()方法，只有在调用了该方法后才能对 ImageSwitcher 对象进行其他的操作；ImageSwitcher 的切换效果由第 37、39 行的方法实现，这里的 setInAnimation()是资源被读入到这个 ImageSwitcher 中时实现的动画效果，setOutAnimation()是资源从这个 ImageSwitcher 中消失的时候要实现的动画效果，本例中使

用的动画效果是从 Android. R 系统文件中读取的。

⑤ 第 54～61 行定义了 makeView()方法，在其中创建了一个 ImageView 对象，设置了该对象的一些属性，并返回给 ImageSwitcher 对象 mySwitcher。

⑥ 第 43 行为 Gallery 对象 g 绑定一个自定义的 ImageAdapter 适配器。该适配器继承自 BaseAdapter，在第 64～78 行定义了这个 ImageAdapter 适配器类，在 70～77 行重写了 getView()方法，为滚动的图片列表索引赋予资源数组 myThmIds 的值。

⑦ 第 44 行为 Gallery 对象 g 添加一个滚动监听 OnItemSelectedListener。在第 47～49 行为该监听定义 OnItemSelected()回调方法，在其中为 ImageSwitcher 对象 mySwitcher 获取选中图片的资源，并显示出来。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_ImageSwitch 项目。运行结果如图 5-19 所示。

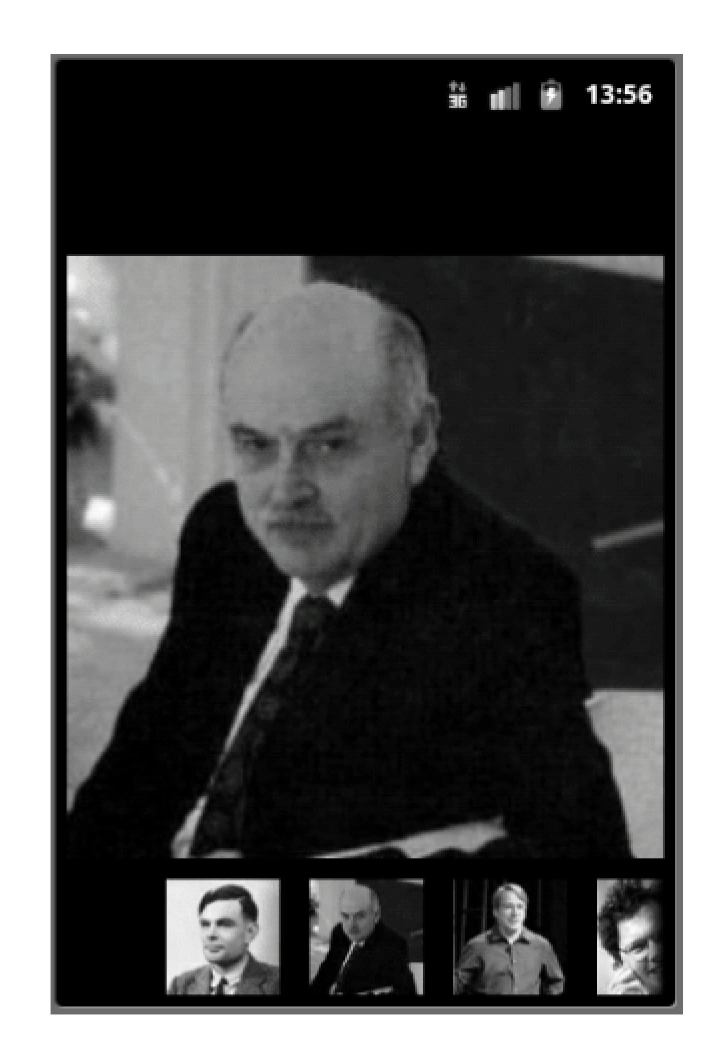


图 5-19 图片浏览切换展示窗的效果

5.2.3 进度条与滑块控件

前面介绍了 Android 中常用的各种视图控件，本节将对 Android 应用中常用的进度条和拖动条进行简单介绍。

1. ProgressBar

ProgressBar 类位于 android. widget 包下，它是一个非常有用的控件，其最直观的效果就是进度条显示。通常在应用程序执行某些较长时间加载资源或执行某些耗时的操作时，会使用到进度条。

ProgressBar 类的使用非常简单，只需要将其显示在前台，然后启动一个后台线程定时更新进度条的进度 progress 数值即可。

2. SeekBar

SeekBar 位于 android. widget 包下，它继承自 ProgressBar，功能相似，不同点在于 SeekBar 是可以被用户拖动的控件。SeekBar 类似于一个尺子，可以进行拖拉滑块，可以直观地显示数据，常用于调节声音大小的应用场合。

3. RatingBar

RatingBar 类位于 android. widget 包下，它是另一种滑块，外观是 5 个星星，可以通过拖动来改变进度。一般用于星级评分的场合。

下面通过一个完整案例来介绍进度条和滑块条的使用方法。

【案例 5.13】 在屏幕中各放置一个 ProgressBar、SeekBar 和 RatingBar，当拖动 SeekBar 时，另外两个跟着同步移动，当拖动 RatingBar 时，另外两个也同步移动。

【说明】 为 ProgressBar 和 SeekBar 设置进度刻度使用方法 setProgress(int)，获得其进

度数据使用方法 getProgress()；而为 RatingBar 设置星级使用方法 setRating(float)，获得其星级使用方法 getRating()。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_ProgressBars 的 Android 项目。其应用程序名为 ProgressBars，包名为 cn. com. sgmsc. progressbars，Activity 组件名为 ProgressBarsActivity。

(2) 设计布局。重命名 res/layout 目录下的 main. xml 文件为 progress_bars. xml，并编辑之。代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3        android:orientation = "vertical"
4        android:layout_width = "fill_parent"
5        android:layout_height = "fill_parent">
6      <TextView
7          android:id = "@ + id/Text01"
8          android:layout_width = "fill_parent"
9          android:layout_height = "wrap_content"
10         android:text = "ProgressBar:"/>
11     <ProgressBar
12         android:id = "@ + id/ProgressBar01"
13         android:layout_width = "fill_parent"
14         android:layout_height = "wrap_content"
15         android:max = "100"
16         android:progress = "20"
17         style = "@android:style/Widget.ProgressBar.Horizontal"/>
18      <TextView
19         android:id = "@ + id/Text02"
20         android:layout_width = "fill_parent"
21         android:layout_height = "wrap_content"
22         android:text = "SeekBar:"/>
23     <SeekBar
24         android:id = "@ + id/SeekBar01"
25         android:layout_width = "fill_parent"
26         android:layout_height = "wrap_content"
27         android:max = "100"
28         android:progress = "20"/>
29      <TextView
30         android:id = "@ + id/Text03"
31         android:layout_width = "fill_parent"
32         android:layout_height = "wrap_content"
33         android:text = "RatingBar:"/>
34     <RatingBar
35         android:id = "@ + id/RatingBar01"
36         android:layout_width = "wrap_content"
37         android:layout_height = "wrap_content"
38         android:max = "5"
39         android:rating = "1"
40     />
41
42 </LinearLayout>
```

第 17 行设置 Progress 进度条为条形进度条。

(3) 开发逻辑代码。打开 src/cn.com.sgmsc.progressbars 包下的 ProgressBarsActivity.java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.progressbars;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ProgressBar;
import android.widget.RatingBar;
import android.widget.SeekBar;

public class ProgressBarsActivity extends Activity {
    final static double MAX = 100;                    //SeekBar、ProgressBar 的最大值
    final static double MAX_STAR = 5;                 //RatingBar 的最大星星数

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.progress_bars);

        //普通拖拉条被拉动的处理代码
        SeekBar sb = (SeekBar)this.findViewById(R.id.SeekBar01);
        sb.setOnSeekBarChangeListener(
            new SeekBar.OnSeekBarChangeListener(){
              public void onProgressChanged(SeekBar seekBar, int progress,
                        boolean fromUser) {
                    ProgressBar pb = (ProgressBar)findViewById(R.id.ProgressBar01);
                    RatingBar rb = (RatingBar)findViewById(R.id.RatingBar01);
                    SeekBar sb = (SeekBar)findViewById(R.id.SeekBar01);
                    pb.setProgress(sb.getProgress());
                    rb.setRating((float)(sb.getProgress()/MAX * MAX_STAR));
                                                       //将进度值折算成星级数
              }
              public void onStartTrackingTouch(SeekBar seekBar) {   }
              public void onStopTrackingTouch(SeekBar seekBar) { }
            }
        );

        RatingBar rb = (RatingBar)findViewById(R.id.RatingBar01);
        rb.setOnRatingBarChangeListener(
              new RatingBar.OnRatingBarChangeListener(){
                  @Override
                  public void onRatingChanged(RatingBar ratingBar, float rating,
                            boolean fromUser) {
                      ProgressBar pb = (ProgressBar)findViewById(R.id.ProgressBar01);
                      SeekBar sb = (SeekBar)findViewById(R.id.SeekBar01);
                      RatingBar rb = (RatingBar)findViewById(R.id.RatingBar01);
                      float rate = rb.getRating();
                      pb.setProgress((int) (rate/MAX_STAR * MAX));
                                        //将 0~5 的星星数折算成 0~100 的进度值
                      sb.setProgress((int) (rate/MAX_STAR * MAX));
                                        //将 0~5 的星星数折算成 0~100 的进度值
                  }
              }
        );

    }
}
```

① 第19～32行设置SeekBar的拖动监听，当SeekBar被拖拉时，同步设置ProgressBar和RatingBar的值。

② 第34～48行设置RatingBar的拖动监听，当RatingBar被拖拉时，同步设置ProgressBar和SeekBar的值。

【运行结果】 在Eclipse中启动Android模拟器，然后运行Activity_ProgressBars项目。运行结果如图5-20所示。

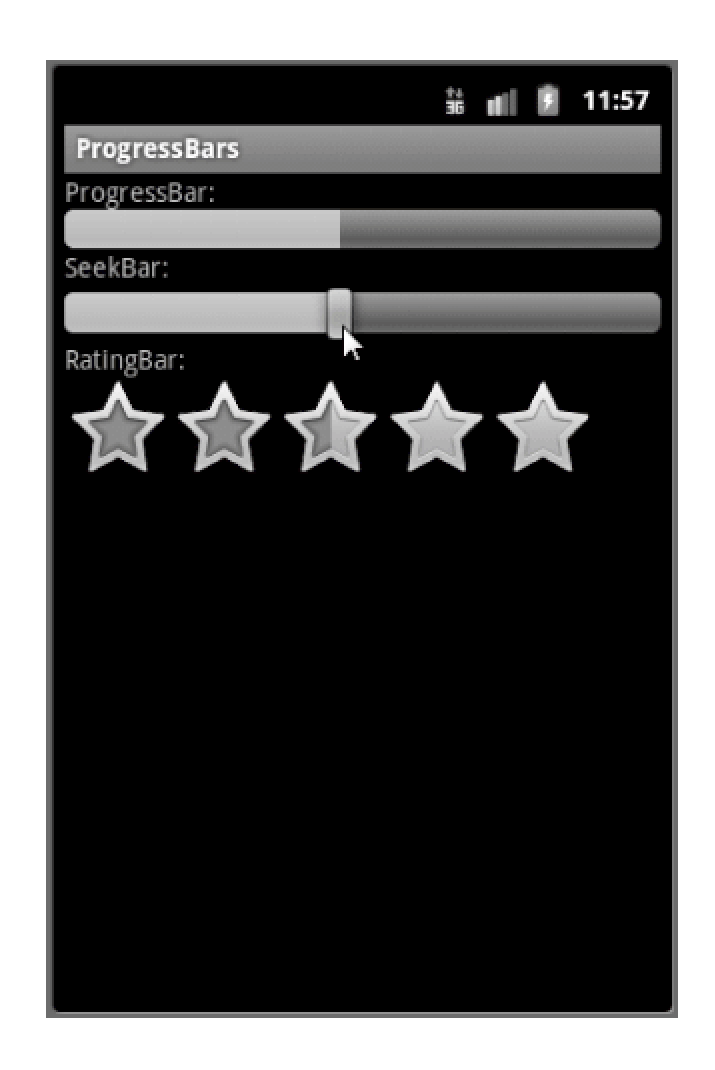


图5-20　滑块/星星被拖拉时与进度条的进度同步效果

5.3　UI设计及应用案例

下面通过一个应用项目的中心框架设计来理解本章中所介绍的内容。

【案例5.14】 用选项卡的形式设计“掌上微博”的中心框架界面，通过这个界面可以跳转到“掌上微博”的任何可操作的功能界面中。

【说明】 可以选用TabHost控件来设计这个中心框架界面。首先根据“掌上微博”的功能模块分类来定义选项页Tab的标题信息。然后在每一个选项页中显示相应的Activity。

在Tab中显示Activity，需要创建一个Intent对象来绑定一个Activity，并使用setContent()方法来载入这个Intent对象。这样就可以完成在一个Tab中显示另一个Activity了。

【开发步骤及解析】

(1) 功能设计。“掌上微博”的功能分为快捷、好友、日志和相册4大模块，其中在“快捷”Tab中，将“掌上微博”中常用的功能用列表的形式显示出来。

(2) Activity的界面设计。根据“掌上微博”的功能可使用TabHost控件，按“快捷”、“好友”、“日志”和“相册”的顺序添加Tab。其中第一个Tab就是默认显示的界面。在本例中，只详细设计这个默认的界面，即“快捷”Activity的功能页，其余的Activity界面将在后续章节逐一补充。这个“快捷”页的界面很简单，它只需要使用一个ListView控件，将常用的功能一一列出即可。在此，省略其草图绘制部分。

(3) 创建项目。在 Eclipse 中创建一个名为 ZSWB_Menu 的 Android 项目。其应用程序名为 ZSWB，包名为 cn. com. sgmsc. ZSWB，Activity 组件名为 FrameTabActivity。

(4) 准备图片。这个项目需要准备一套图标，用于在 TabHost 的标签中和 ListView 的条目中显示，将制作的图片资源复制到本项目的 res/drawable-mdpi 目录中。

(5) 准备字符串资源。编写 res/values 目录下的 strings. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "hello">Hello World, ZSWB!</string>
    <string name = "app_name">掌上微博</string>
    <string name = "ZSWB">掌上微博</string>

    <string name = "PublishActivity">掌上微博 -- 快速发布</string>
    <string name = "ContactsActivity">掌上微博 -- 最近访客</string>
    <string name = "MyDiaryActivity">掌上微博 -- 我的日志</string>
    <string name = "AlbumListActivity">掌上微博 -- 相册列表</string>
    <string name = "btnBack">返回</string>
    <string name = "tabtitle1">快捷</string>
    <string name = "tabtitle2">好友</string>
    <string name = "tabtitle3">日志</string>
    <string name = "tabtitle4">相册</string>

</resources>
```

(6) 创建数组资源。创建并编写 res/values 目录下的数组描述文件 arrays. xml，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string - array name = "lvtexts">
        <item>写书日志</item>
        <item>拍照上传</item>
  </string - array>
  <string - array name = "lvicons">
        <item>p_diary</item>
        <item>p_camera</item>
    </string - array>

</resources>
```

(7) 创建颜色资源。创建并编写 res/values 目录下的颜色描述文件 colors. xml，参见本书源代码，源代码可从指定网站中下载，在此不作赘述。

(8) 创建样式资源。创建并编写 res/values 目录下的样式描述文件 style. xml，参见本书源代码。

(9) 设计初始进入的 Activity 布局。重命名 res/layout 目录下的 main. xml 文件为 normal. xml，并编辑之。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
 xmlns:android = "http://schemas.android.com/apk/res/android"
 android:orientation = "vertical"
 android:background = "@drawable/back"
```

```
6    android:layout_width = "fill_parent"
7    android:layout_height = "fill_parent"
8    >
9    <ListView
10           android:id = "@ + id/lvPublish"
11           android:divider = "@color/listDivider"
12           android:dividerHeight = "2px"
13           android:layout_width = "fill_parent"
14           android:layout_height = "fill_parent"
15           />
16  </LinearLayout>
```

第 11～12 行为 ListView 控件的分隔线设置颜色和线的高度。

（10）设计其他 Activity 布局。创建并编写 res/layout 目录下的其他布局文件 contacts.xml、diary_list.xml、album_list.xml。本例不详细开发这三个文件的布局，只是分别在 diary_list.xml、album_list.xml 文件中添加一个 Button 而已，下面是 diary_list.xml 文件的代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout
 xmlns:android = "http://schemas.android.com/apk/res/android"
 android:orientation = "vertical"
 android:paddingLeft = "8px"
 android:background = "@drawable/back"
 android:layout_width = "fill_parent"
 android:layout_height = "fill_parent"
 >
 <ScrollView
        android:fillViewport = "true"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_alignParentTop = "true"
        android:layout_alignParentLeft = "true"
        >
        <TextView
              android:id = "@ + id/txtv"
              android:layout_width = "fill_parent"
              android:layout_height = "fill_parent"
              android:textColor = "@color/character"
              android:text = " 日志页面建设中..."
              android:textSize = "20dip"
             />
 </ScrollView>
 <Button
        android:id = "@ + id/Dia_btnBack"
        style = "@style/button"
        android:layout_width = "120px"
        android:layout_height = "wrap_content"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentRight = "true"
        android:layout_marginRight = "5px"
        android:text = "@string/btnBack"
        />
</RelativeLayout>
```

(11) 开发项目的第一个 Activity。打开 src/cn.com.sgmsc.ZSWB 包下的 FrameTabActivity.java 文件，并编辑之。这个文件是对本项目的 TabHost 控件进行逻辑代码设计，可以认为是本应用项目的中心框架部分，代码如下所示。

```
1  package cn.com.sgmsc.ZSWB;
2
3  import android.app.TabActivity;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.view.Window;
7  import android.widget.TabHost;
8
9  public class FrameTabActivity extends TabActivity{
10 @Override
11 protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13         requestWindowFeature(Window.FEATURE_NO_TITLE);  //设置这个 Activity 没有标题栏
14         final TabHost tabHost = getTabHost();
15         Intent itNormal = new Intent(this,NormalActivity.class);
16         Intent itContact = new Intent(this,ContactsActivity.class);
17         Intent itDiary = new Intent(this,DiaryListActivity.class);
18         Intent itAlbum = new Intent(this,AlbumListActivity.class);
19
20         tabHost.addTab(tabHost.newTabSpec("tab1")
21                 .setIndicator(getResources().getText(R.string.tabtitle1),
22                         getResources().getDrawable(R.drawable.publish))
23                 .setContent(itNormal)
24                 );
25         tabHost.addTab(tabHost.newTabSpec("tab2")
26                 .setIndicator(getResources().getText(R.string.tabtitle2),
27                         getResources().getDrawable(R.drawable.friend))
28                 .setContent(itContact)
29                );
30         tabHost.addTab(tabHost.newTabSpec("tab3")
31                 .setIndicator(getResources().getText(R.string.tabtitle3),
32                         getResources().getDrawable(R.drawable.diary))
33                 .setContent(itDiary)
34                 );
35         tabHost.addTab(tabHost.newTabSpec("tab4")
36                 .setIndicator(getResources().getText(R.string.tabtitle4),
37                         getResources().getDrawable(R.drawable.album))
38                 .setContent(itAlbum)
39                 );
40        }
41 }
```

① 第 13 行设置窗口的标题栏不显示。使用此设置需要引入 android.view.Window 包。

② 第 15～18 行声明 4 个 Intent 对象，分别绑定 4 个 Activity。

③ 从第 20 行起逐一向 TabHost 对象 tabHost 中添加选项卡。其中第 20～24 行是添加名为“tab1”的选项卡，在 setIndicator 中使用 getResources().getText(R.string.tabtitle1)设置 Tab 的标题文本，使用 getResources().getDrawable(R.drawable.publish)设置 Tab 的标题图标；使用 setContent 设置该选项卡显示的内容，这里指定的是 itNormal(Intent 对象)，即显示 NormalActivity 的界面。

(12) 开发 TabHost 中第一个 Tab 对应的 Activity。创建并编写 src/cn.com.sgmsc.ZSWB 包下的 NormalActivity.java 文件，该文件是 TabHost 中第一个 Tab 所对应的 Activity，完成本项目中对快捷功能模块的逻辑实现。代码如下所示。

```
package cn.com.sgmsc.ZSWB;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;

public class NormalActivity extends Activity{
@Override
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.normal);                              //设置当前屏幕
      ListView lvPublish = (ListView)findViewById(R.id.lvPublish); //获得 ListView 对象引用

      BaseAdapter ba = new BaseAdapter() {

          //取得 ListView 的 item 项文本内容的数组
          String[] items = getResources().getStringArray(R.array.lvtexts);
          //取得 ListView 的 item 项图标资源的数组
          String[] imgIds = getResources().getStringArray(R.array.lvicons);

          @Override
          public View getView(int position, View convertView, ViewGroup parent) {
                 LinearLayout ll = new LinearLayout(NormalActivity.this);
                 ll.setOrientation(LinearLayout.HORIZONTAL);
                 ll.setGravity(Gravity.CENTER);
                 ImageView iv = new ImageView(NormalActivity.this);
                                                                    //创建 ImageView 对象
                 iv.setAdjustViewBounds(true);

                 iv.setImageDrawable(getResources().getDrawable(
                           getResources().getIdentifier(imgIds[position],"drawable",
                           getPackageName())));
                 ll.addView(iv);                           //将 ImageView 添加到线性布局中
                 TextView tv = new TextView(NormalActivity.this);
                 tv.setPadding(10, 0, 0, 0);
                 tv.setLayoutParams(new LinearLayout
                        .LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
                 tv.setTextAppearance(NormalActivity.this, R.style.title);
                 tv.setText(items[position]);                       //设置 TextView 显示的内容
                 ll.addView(tv);
```

```
50                return ll;
51            }
52            @Override
53            public long getItemId(int position) {
54                return 0;
55            }
56            @Override
57            public Object getItem(int position) {
58                return null;
59            }
60            @Override
61            public int getCount() {
62                return items.length;
63            }
64        };
65
66        lvPublish.setAdapter(ba);
67
68        lvPublish.setOnItemClickListener(new OnItemClickListener() {
69            @Override
70            public void onItemClick(AdapterView<?> parent, View v, int position,
71                    long id) {
72                switch(position){                          //判断选择的是哪一个选项
73                case 0:                                    //书写日志
74                    Intent intentDia = new Intent();
75                    intentDia.setClass(NormalActivity.this, DiaryListActivity.class);
76                    startActivity(intentDia);
77                    break;
78                case 1:                                    //拍照上传
79                    Intent intentAlb = new Intent();
80                    intentAlb.setClass(NormalActivity.this, AlbumListActivity.class);
81                    startActivity(intentAlb);
82                    break;
83                }
84            }
85        });
86    }
87 }
```

① 第 28 行和第 30 行从数组描述文件中取得 ListView 的 item 项显示的文本和图标数据。

② 第 32～51 行重写 BaseAdapter 的 getView()方法，定义 ListView 对象 lvPublish 每一个 item 的显示及内容。

③ 第 68～86 行为 lvPublish 对象添加 OnItemClickListener 监听，当单击了 lvPublish 中的条目时，会跳转到相应的 Activity 中去。

（13）开发其余的 Activity。创建并编写 src/cn.com.sgmsc.ZSWB 包下的 ContactsActivity.java、DiaryListActivity.java 和 AlbumListActivity.java 文件。本例不详细开发这三个代码文件，只是分别对 DiaryListActivity.java 和 AlbumListActivity.java 文件中的 Button 监听进行了简单的事件回调处理而已，在此以 DiaryListActivity.java 文件为例，给出其代码如下所示。

```
package cn.com.sgmsc.ZSWB;

import android.app.Activity;
import android.content.Intent;
```

```
5  import android.os.Bundle;
6  import android.view.View;
7  import android.view.View.OnClickListener;
8  import android.widget.Button;
9
10 public class DiaryListActivity extends Activity{
11    @Override
12    protected void onCreate(Bundle savedInstanceState) {
13        super.onCreate(savedInstanceState);
14        setContentView(R.layout.diary_list);          //设置当前屏幕
15
16        Button btnback = (Button) findViewById(R.id.Dia_btnBack);
17        OnClickListener btnlsn = new OnClickListener(){
18               @Override
19               public void onClick(View v) {
20                     Intent intentback = new Intent();
21                     intentback.setClass(DiaryListActivity.this, FrameTabActivity.class);
22                     startActivity(intentback);
23                     finish();
24               }
25         };
26         btnback.setOnClickListener(btnlsn);
27    }
28
29 }
```

① 第 19～24 行定义 onClick()回调方法，当单击了按钮时，会跳转回 FrameTabActivity 中。

② 第 23 行的 finish()方法是用来关闭 DiaryListActivity。注意，当调用 finish()方法时，只是将 Activity 推向后台，并没有立即释放内存，Activity 的资源并没有被清理；而当调用 System.exit(0)时，是真正杀死整个进程，这时候 Activity 所占的资源也随之被释放。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 KDWB_Menu 项目。初始运行效果如图 5-21 所示。选择“日志”选项卡后如图 5-22 所示，这时单击“返回”按钮，可以回到初始界面。在初始界面中单击“书写日志”项进入写日志页，如图 5-23 所示。



图 5-21 初始运行时的界面效果



图 5-22 选择“日志”选项卡后的界面



图 5-23 在初始界面单击“书写日志”后的界面

小结

本章较详细地介绍了 Android 事件的处理机制，包括几种常用的回调事件和基于监听接口的事件处理。接下来，重点介绍了与适配器相关的控件 AutoCompleteTextView、Spinner、ListView、GridView 和 Gallery，还简单地介绍了其他视图控件和进度条与滑块控件。对于所介绍的控件都给出了完整案例，从多个角度说明了常见适配器的构建方法和常用监听接口的添加与回调方法的编程。

本章的案例较多，并且都有一定的深度。相信读者在学习之后能较好地掌握 Android 的控件应用编程。这些控件以及第 4 章中介绍的控件都是 Android 应用程序中经常用到的屏幕元素，熟练地掌握它们的使用方法是必不可少的技能。当然，在实际开发中只运用这些控件是不够的，还有一些控件如菜单、对话框，也是应用程序经常使用的屏幕元素，将在第 6 章中介绍。

练习

1. 使用 ScrollView 控件，将第 4 章、第 5 章的控件使用案例组合到一个应用项目中，要求在屏幕中垂直放置多个按钮，单击每个按钮，对应调用一个案例的 Activity。

提示一下：

(1) ScrollView 控件内只能包含一个控件，可以在项目的 main. xml 布局文件中如下设计。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< ScrollView  …>
     < LinearLayout …>
         < Button … />
         < Button … />
         < Button … />
         ……
         < Button … />
     </LinearLayout …>
</ScrollView  …>
```

(2) 其他的布局文件和 Java 代码文件可以从相应的案例中复制过来，同时别忘记有些案例中还包括相关资源如图片资源、数组资源、strings. xml、colors. xml 等，都要一并复制过来。

(3) 这个项目中有多个 Activity，一定要在 AndroidManifest. xml 中声明之。

2. 在第 4 章完成的微博登录界面和注册界面设计的基础上，增加如下功能：单击登录界面中的“注册”按钮，可跳转到注册界面中，单击注册界面中的“返回”按钮，可返回到登录界面。

第6章 菜单与对话框

在实际的应用开发中经常要用到菜单和对话框。在应用程序中添加一些菜单，或在用户执行了某些操作后给出一些对话框回馈信息，可以让用户体验更完善，让程序功能更完备。

6.1 菜单

菜单(Menu)的设计在人机交互中可以说是非常人性化的，它提供了不同功能分组展示的能力。Android 平台提供三种类型的菜单：选项菜单(OptionMenu)，上下文菜单(ContextMenu)和子菜单(SubMenu)。

6.1.1 选项菜单

当用户单击手机设备上的 Menu 按键时，弹出的菜单就是选项菜单(OptionMenu)。选项菜单的菜单项最多只能有 6 个，每个菜单项可以携带图标。当菜单项超过 6 个时，第 6 个自动显示“更多”选项来展示显示，且这个菜单项将不会显示图标。Android 系统自带有选项菜单，在其模拟器中，当单击 Menu 按键时，将会在屏幕底部弹出如图 6-1 所示的选项菜单。

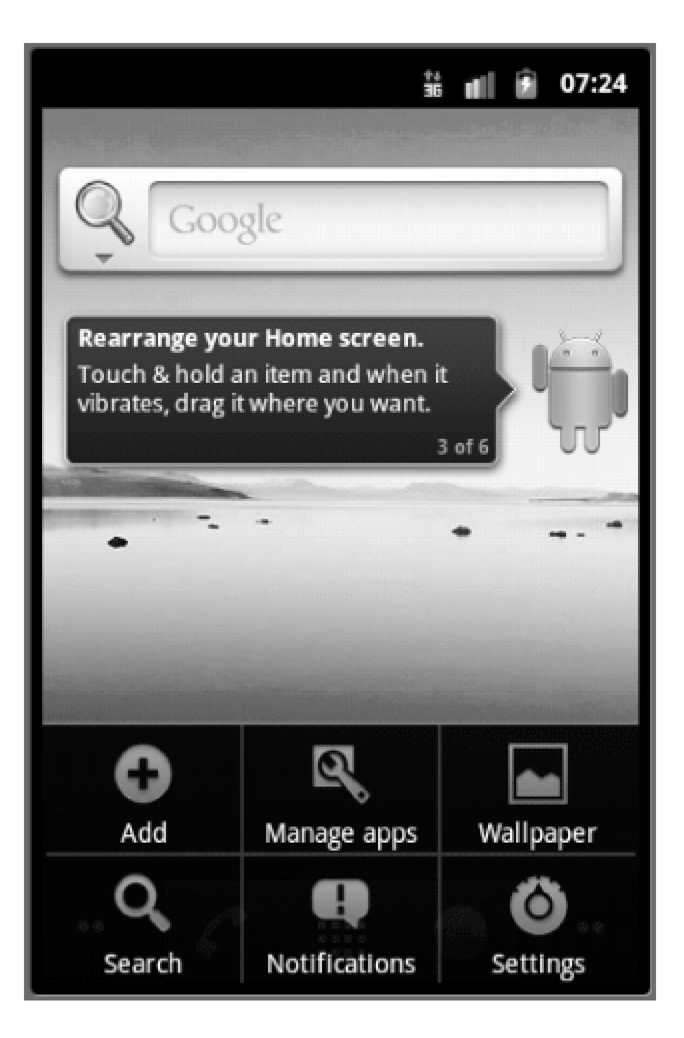


图 6-1 Android 模拟器自带的选项菜单

OptionMenu 在 android. view. Menu 包中定义，一个选项菜单是一个 Menu 对象，Menu 对象中可以添加菜单项 MenuItem。选项菜单的功能需要开发人员编程来实现，如果在开发应用程序中没有实现选项菜单功能，那么程序运行时按下手机上的 Menu 键是不会起作用的。在 Android 中通过回调方法创建菜单并处理菜单按下的事件。选项菜单常用的回调方法如表 6-1 所示。

表 6-1 选项菜单中常用的回调方法及说明

方法	描述
onCreateOptionsMenu(Menu)	初始化选项菜单，只在首次显示菜单时调用
onOptionsItemSelected(MenuItem)	当其中某个菜单项被选中时调用，默认返回 false
onOptionsMenuClosed(Menu)	当选项菜单关闭、或按下返回键、或选择了某菜单项时调用
onPrepareOptionsMenu(Menu)	为程序准备选项菜单，每次选项菜单显示前调用

Menu 类对象是一个菜单，它包含一个或多个菜单项 MenuItem，也可以包含子菜单 SubMenu。Menu 中常用的方法如表 6-2 所示。

表 6-2　Menu 中常用的方法及说明

方　法	参　数	描　述
Add(int groupId, int itemId, int order, CharSequence title)	groupId：菜单项所在的组 Id，通过分组可以对菜单项进行批量操作，如果菜单项不属于任何组，传入 NONE； itemId：菜单项的 ID； order：菜单项的顺序； title：菜单项显示的文本对象； titleRes：菜单项显示文本的资源符	向 Menu 中添加一个菜单项，返回一个 MenuItem 对象
Add(int groupId, int itemId, int order, int titleRes)		
Add(CharSequence title)		
Add(int titleRes)		
finditem(int Id)	Id 是 MenuItem 对象的标识符	返回指定 Id 的 MenuItem 对象
size()		返回 Menu 中菜单项的个数

MenuItem 对象代表一个菜单的菜单项，当用户选择菜单项时需要一些回调方法给予响应。常用的回调方法如表 6-3 所示。

表 6-3　MenuItem 中常用的方法及说明

方　法	参　数	描　述
setAlphabeticShortcut(char alphaChar)	alphaChar：字母快捷键	设置 MenuItem 的字母快捷键
setNumericShortcut(char numericChar)	numericCh：数字快捷键	设置 MenuItem 的数字快捷键
setIcon(Drawable icon)	Icon：图标对象	设置 MenuItem 的图标
setIntent(Intent intent)	Intent：与 MenuItem 绑定的 Intent 对象	为 MenuItem 绑定 Intent 对象，当被选中时将调用 startActivity 方法
setOnMenuItemClickListener (MenuItem. OnMenuItemClickListener menuItemClickListener)	menuItemClickListener：监听器	为 MenuItem 设置监听器，回调方法一般采用 onOptionsItemSelected()
setShortcut(char numericChar, char alphaChar)	numericChar：数字快捷键 alphaChar：字母快捷键	设置 MenuItem 的数字快捷键和字母快捷键
setTitle(int titleRes)	titleRes：标题文本的资源符	设置 MenuItem 的标题
setTitle(CharSequence title)	title：标题文本对象	

设计一个选项菜单时要为用户提供交互接口，以响应菜单项被单击的事件。创建选项菜单需要如下步骤。

(1) 第一步，重写 Activity 的 onCreateOptionsMenu(Menu menu)方法，当第一次打开菜单时该方法被自动调用。

(2) 第二步，调用 Menu 的 add()方法添加菜单项 MenuItem，此时，可以调用 MenuItem 的 setIcon()方法来为菜单项设置图标。

(3) 第三步，定义菜单项被选择之后的回调事件。有两种方法：其一，重写 Activity 的 onOptionsItemSelected()方法，当菜单项 MenuItem 被选择时，该方法用于响应事件；其二，为每个菜单项 MenuItem 对象添加 OnMenuItemClickListener 监听器，在其中定义处理菜单选项中的事件。

下面通过一个简单案例来说明选项菜单的使用方法。

【案例 6.1】 设置选项菜单，其中有两个菜单项："开始"和"返回"。当接收用户在菜单中的选项后，在屏幕的文本框控件中显示选择的内容。

【说明】 在本例中，使用为菜单项 MenuItem 对象添加 OnMenuItemClickListener 监听器的方式处理选择事件。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_OptionMenu 的 Android 项目。其应用程序名为 OptionMenu，包名为 cn.com.sgmsc.optionmenu，Activity 组件名为 OptionMenuActivity。

(2) 准备图片。将图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 设计布局。编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent"
     android:orientation = "vertical" >

     < TextView
         android:id = "@ + id/tv01"
         android:layout_width = "fill_parent"
         android:layout_height = "wrap_content"
         android:textSize = "26px"
         android:text = "请选择..." />

</LinearLayout >
```

(4) 开发逻辑代码。打开 src/cn.com.sgmsc.optionmenu 包下的 OptionMenuActivity.java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.optionmenu;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MenuItem.OnMenuItemClickListener;

public class OptionMenuActivity extends Activity {

     private TextView text;
     //菜单项 ID 常量
     private static final int ITEM1 = Menu.FIRST;
     private static final int ITEM2 = Menu.FIRST + 1;

     @Override
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.main);

         text = (TextView) findViewById(R.id.tv01);
     }
```

```
24
25      //添加菜单项
26      public boolean onCreateOptionsMenu(Menu menu) {
27          //添加两个菜单项
28
29          MenuItem begmenuitem = menu.add(0, ITEM1, 0, "开始");
30          begmenuitem.setIcon(R.drawable.ic_enter);
31          MenuItem retmenuitem = menu.add(0, ITEM2, 0, "返回");
32          retmenuitem.setIcon(R.drawable.ic_return);
33
34          OnMenuItemClickListener lsn = new OnMenuItemClickListener(){
35              //响应菜单选项被单击事件
36              @Override
37              public boolean onMenuItemClick(MenuItem item){
38              switch (item.getItemId()) {
39              //
40              case ITEM1:
41                  text.setText("你选择了：开始。");
42                  break;
43              case ITEM2:
44                  text.setText("你选择了：返回。");
45                  break;
46              }
47              return true;
48              }
49          };
50
51          begmenuitem.setOnMenuItemClickListener(lsn);
52          retmenuitem.setOnMenuItemClickListener(lsn);
53          return true;
54      }
55
56 }
```

① 第 14～15 行定义菜单项 ID 常量，用于稍后设置菜单项使用。

② 第 26～49 行定义了 onCreateOptionsMenu ()方法，在其中为菜单添加了两个菜单项，第 29 行添加了菜单项 begmenuitem，其 ID 号为常量 ITEM1，其显示的文本为“开始”，第 30 行为此菜单项设置了图标；同样，第 31～32 行添加了另一个菜单项。

③ 在 onCreateOptionsMenu()方法内还创建了 OnMenuItemClickListener 监听对象 lsn，并定义了回调方法 onMenuItemClick()。其中使用 switch 语句定义：当选择菜单 ID 为 ITEM1 时，屏幕的 Text View 控件显示“你选择了：开始。”，当选择菜单 ID 为 ITEM2 时，屏幕的 TextView 控件显示“你选择了：返回。”

④ 第 51、52 行为两个菜单项添加 OnMenuItemClickListener 监听。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_OptionMenu 项目。初始运行时，当按下手机（或模拟器）中的 menu 键时，屏幕显示如图 6-2 所示，当按下了“开始”菜单项时，屏幕显示如图 6-3 所示，放开按下的菜单项，则屏幕显示如图 6-4 所示。

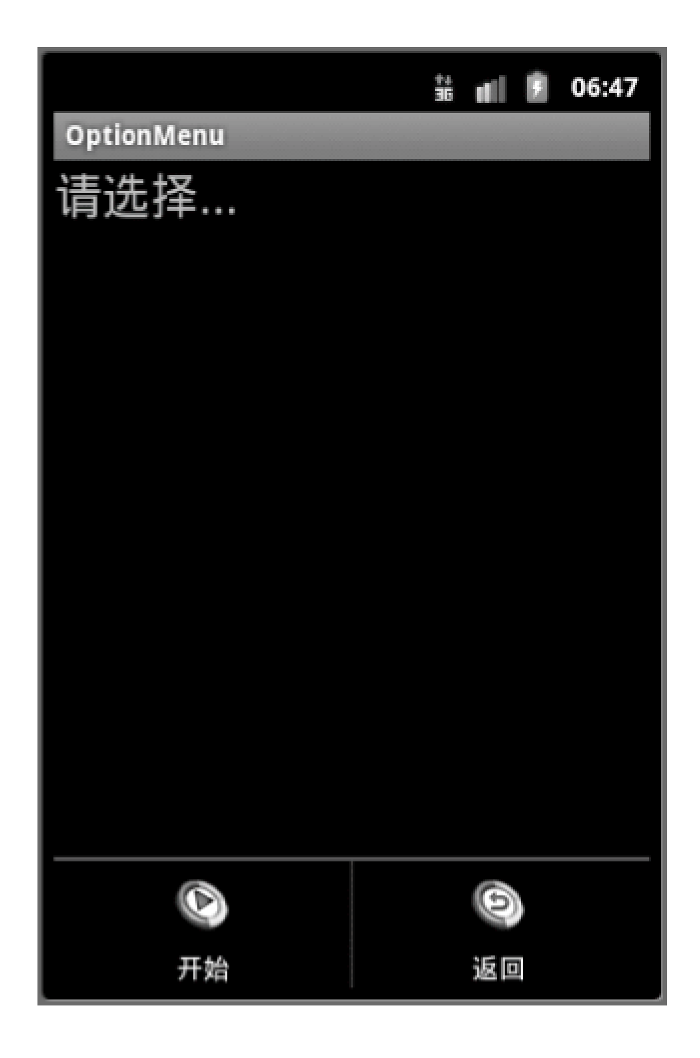


图 6-2　按下手机的 menu 键，显示选项菜单



图 6-3　按下“开始”菜单项的显示效果



图 6-4　释放“开始”菜单项的显示效果

6.1.2　子菜单

子菜单(SubMenu)类位于 android. view 包下，它继承自 Menu，每个 SubMenu 对象代表一个子菜单。子菜单就是将相同功能的菜单项分组进行多级显示的一种菜单，例如，Word 窗口中的“文件”菜单中有“新建”、“打开”、“关闭”等子菜单。

SubMenu 通常与选项菜单联合使用，往菜单中添加子菜单与添加菜单项的方法差不多，只是方法名不一样，使用 addSubMenu( ) 方法，但其参数与 add( ) 方法中的参数一样。SubMenu 中常用的方法如表 6-4 所示。

表 6-4　SubMenu 中常用的方法及说明

方　法	参　数	描　述
setHeaderIcon(Drawable icon)	icon：图标对象	设置子菜单的标题图标
setHeaderIcon(int id)	id：图标资源 ID	
setHeaderTitle(CharSequence title)	title：文本对象	设置子菜单的标题文本内容
setHeaderTitle(int id)	id：文本资源 ID	
setHeaderView(View view)	view：标题的 View 对象	设置指定的 View 对象作为子菜单的图标
setIcon(Drawable icon)	icon：图标对象	设置子菜单在父菜单中显示的图标
setIcon(int id)	id：图标资源 ID	

下面通过一个案例来说明选项菜单与子菜单的用法。

【案例 6.2】 设置选项菜单，其中有两个菜单分别是：“性别”子菜单，“爱好”子菜单。当接收用户选择了子菜单中的菜单项时，在屏幕的文本编辑框控件中累计记录所做的选择内容。

【说明】 在本例中，“性别”子菜单选用单选子菜单方式构建，“爱好”子菜单选用复选子菜单方式构建。处理菜单项被选择的响应事件，采用重写 Activity 的 onOptionsItemSelected( ) 方法。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_Menu 的 Android 项目。其应用程序名为“Menu and SubMenu”，包名为 cn. com. sgmsc. Menu，Activity 组件名为 MenuActivity。

(2) 准备图片。将图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 准备字符串资源。编写 res/values 目录下的 strings. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "hello">Hello World, MenuActivity!</string>
    <string name = "app_name">Menu and SubMenu</string>
    <string name = "label">您的选择为: \n</string>
    <string name = "gender">性别</string>
    <string name = "male">男</string>
    <string name = "female">女</string>
    <string name = "hobby">爱好</string>
    <string name = "hobby0">运动</string>
    <string name = "hobby1">唱歌</string>
    <string name = "hobby2">旅游</string>
    <string name = "hobby3">游戏</string>
    <string name = "hobby4">冒险</string>
    <string name = "hobby5">摄影</string>
</resources>
```

(4) 设计布局。编写 res/layout 目录下的 main. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
      android:id = "@ + id/LinearLayout01"
      android:background = "@drawable/bg02"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent"
      android:orientation = "vertical"
      xmlns:android = "http://schemas.android.com/apk/res/android"><! -- 声明一个线性
      布局 -->
    <ScrollView
      android:id = "@ + id/ScrollView01"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent">          <! -- 声明一个 ScrollView 控件 -->
        <EditText
            android:id = "@ + id/EditText01"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:editable = "false"
            android:cursorVisible = "false"
            android:text = "@string/label">          <! -- 声明一个 EditText 控件 -->
        </EditText>
    </ScrollView>
</LinearLayout>
```

考虑到将在这个 EditText 控件中不断添加文本内容，可能会超过屏幕的显示范围，所以将这个 EditText 控件放在 ScrollView 控件中。

(5) 开发逻辑代码。打开 src/cn. com. sgmsc. Menu 包下的 MenuActivity. java 文件，并编辑之。代码如下所示。

```
package cn.com.sgms.Menu;

import android.app.Activity;                    //引入相关类
import android.os.Bundle;                       //引入相关类
import android.view.Menu;                       //引入相关类
import android.view.MenuItem;                   //引入相关类
import android.view.SubMenu;                    //引入相关类
import android.widget.EditText;                 //引入相关类

public class MenuActivity extends Activity {
    final int MENU_GENDER_MALE = 0;             //性别为男选项编号
    final int MENU_GENDER_FEMALE = 1;           //性别为女选项编号
    final int MENU_HOBBY0 = 2;                  //爱好 1 选项编号
    final int MENU_HOBBY1 = 3;                  //爱好 2 选项编号
    final int MENU_HOBBY2 = 4;                  //爱好 3 选项编号
    final int MENU_HOBBY3 = 5;                  //爱好 4 选项编号
    final int MENU_HOBBY4 = 6;                  //爱好 5 选项编号
    final int MENU_HOBBY5 = 7;                  //爱好 6 选项编号
    final int MENU_GENDER = 8;                  //"性别"子菜单编号
    final int MENU_HOBBY = 9;                   //"爱好"子菜单编号
    final int GENDER_GROUP = 0;                 //"性别"子菜单项组的编号
    final int HOBBY_GROUP = 1;                  //"爱好"子菜单项组的编号
    final int MAIN_GROUP = 2;                   /外层总菜单项组的编号

    MenuItem[] myHobby = new MenuItem[6];       //"爱好"菜单项组
    MenuItem male = null;                       //男性性别菜单项

    @Override
    public void onCreate(Bundle savedInstanceState) {//重写 onCreate()方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);          //设置当前屏幕
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu){
      //"性别"单选菜单项组    菜单若编组就是单选菜单项组
     SubMenu subMenuGender  =  menu.addSubMenu(MAIN_GROUP,MENU_GENDER,0,R.string.gender);
     subMenuGender.setIcon(R.drawable.gender);
     subMenuGender.setHeaderIcon(R.drawable.gender);
     male = subMenuGender.add(GENDER_GROUP, MENU_GENDER_MALE, 0, R.string.male);
     male.setChecked(true);
     subMenuGender.add(GENDER_GROUP, MENU_GENDER_FEMALE, 0, R.string.female);
    //设置 GENDER_GROUP 组是可选择的,互斥的
     subMenuGender.setGroupCheckable(GENDER_GROUP, true,true);

    //"爱好"复选菜单项组
     SubMenu subMenuHobby  =  menu.addSubMenu(MAIN_GROUP,MENU_HOBBY,0,R.string.hobby);
     subMenuHobby.setIcon(R.drawable.hobby);
     subMenuHobby.setHeaderIcon(R.drawable.hobby);
     myHobby[0] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY0, 0, R.string.hobby0);
     myHobby[1] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY1, 0, R.string.hobby1);
     myHobby[2] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY2, 0, R.string.hobby2);
     myHobby[3] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY3, 0, R.string.hobby3);
     myHobby[4] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY4, 0, R.string.hobby4);
     myHobby[5] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY5, 0, R.string.hobby5);
     myHobby[0].setCheckable(true);             //设置菜单项为复选菜单项
     myHobby[1].setCheckable(true);
     myHobby[2].setCheckable(true);
```

```
59        myHobby[3].setCheckable(true);
60        myHobby[4].setCheckable(true);
61        myHobby[5].setCheckable(true);
62
63        return true;
64      }
65
66      @Override                              //单选或复选菜单项选中状态变化后的回调方法
67      public boolean onOptionsItemSelected(MenuItem mi){
68        switch(mi.getItemId()){
69            case MENU_GENDER_MALE:           //单选菜单项状态的切换要自行写代码完成
70            case MENU_GENDER_FEMALE:
71                  mi.setChecked(true);
72                  appendStateStr();          //当有效项目变化时记录在文本区中
73                  break;
74            case MENU_HOBBY0:                //复选菜单项状态的切换要自行写代码完成
75            case MENU_HOBBY1:
76            case MENU_HOBBY2:
77            case MENU_HOBBY3:
78            case MENU_HOBBY4:
79            case MENU_HOBBY5:
80                  mi.setChecked(!mi.isChecked());
81                  appendStateStr();          //当有效项目变化时记录在文本区中
82                  break;
83        }
84        return true;
85      }
86
87      //获取当前选择状态的方法
88      public void appendStateStr(){
89        String mychoice = "您选择的性别为: ";
90        if(male.isChecked()){
91            mychoice = mychoice + "男";
92        }
93        else{
94             mychoice = mychoice + "女";
95        }
96        //获取被选中的所有爱好
97        String hobbyStr = "";
98        for(MenuItem mi:myHobby){
99             if(mi.isChecked()){
100                 hobbyStr = hobbyStr + mi.getTitle() + "、";
101              }
102         }
103          if(hobbyStr.length()> 0){
104                mychoice = mychoice + ",您的爱好为: " + hobbyStr.substring(0, hobbyStr.length()
                 - 1) + "。\n";
105         }
106         else{
107              mychoice = mychoice + "。\n";
108         }
109         EditText et = (EditText)MenuActivity.this.findViewById(R.id.EditText01);
110         et.append(mychoice);
111       }
112 }
```

① 第 11～12 行定义“性别”子菜单项的 ID 常量；第 13～18 行定义“爱好”子菜单项的 ID 常量；第 19、20 行定义两个子菜单本身的 ID 常量；第 21、22 行定义两个子菜单的分组 ID 常

量；第 23 行定义最外层总菜单组的 ID 常量。

② 第 25 行声明了一个 MenuItem 数组 myHobby，用于存储爱好菜单项组。第 26 行声明了一个菜单项变量 male，用于判断性别，如果 male 为选中状态，表示是“男”性，否则为“女”性。

③ 第 35～64 行定义了菜单创建方法 onCreateOptionsMenu(Menu menu)。其中，第 36～44 行是为“性别”单选菜单创建实例，第 37～39 行是定义“性别”子菜单 subMenuGender，第 40 行为子菜单添加一个菜单项，且赋予菜单项变量 male 中，第 41 行设置该菜单项为选中状态，即设置单选菜单的默认选项，第 44 行设置 GENDER_GROUP 组是可选择的，互斥的，即设置了该子菜单为单选菜单；第 47～61 行是为“爱好”复选菜单创建实例，第 47～49 行是定义“爱好”子菜单 subMenuHobby，第 50～55 行为“爱好”子菜单添加 6 个菜单项，且赋予菜单项数组 myHobby 中，第 56～61 行是设置菜单项为可选的，即设置了该子菜单为复选菜单；第 63 行执行返回定义好的菜单实例，完成 onCreateOptionsMenu(Menu menu)方法。

④ 第 67～85 行定义了 onOptionsItemSelected()方法。在该方法内，使用 switch-case 语句，分别对每一个子菜单项执行操作，第 69～73 行，对于选中的菜单项执行自定义方法 appendStateStr()，第 74～82 行，对于与原选择状态不同的菜单项(即选择状态发生了改变的菜单项的)执行自定义方法 appendStateStr()。

⑤ 第 88～111 行定义 appendStateStr()方法。第 90～95 行，判断 male 的值为 true 时，表示是“男”性，否则为“女”性；第 97～102 行，对 myHobby 数组中的每一元素逐一进行检查，对于处于被选中状态的项，将其菜单项文本内容拼接到字符串变量 hobbyStr 中；第 103～108 行将字符串变量 hobbyStr 中的内容拼接到字符串变量 mychoice 中；第 109～110 行将字符串变量 mychoice 中的内容设置到 EditText 控件 et 对象中，并显示出来。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_Menu 项目。初始运行时，当按下手机(或模拟器)中的 menu 键时，屏幕显示如图 6-5 所示，当按下了“性别”菜单项后屏幕显示如图 6-6 所示，选择了菜单项后屏幕显示如图 6-7 所示，当按下了“爱好”菜单项后屏幕显示如图 6-8 所示，选择了菜单项后屏幕显示如图 6-9 所示。这里，复选菜单项一次只能选择一次，如果要选择多项就需要按多次 menu 键。



图 6-5　按下手机的 menu 键，显示选项菜单



图 6-6　按下“性别”菜单的显示效果



图 6-7　选择“性别”菜单项后的显示效果



图 6-8 按下“爱好”菜单的显示效果



图 6-9 选择“爱好”中两项后的显示效果

6.1.3 上下文菜单

当用户长时间按键不放时，弹出的菜单称为上下文菜单(ContextMenu)。人们经常在手机的文本编辑框设置输入法时会使用上下文菜单 ContentMenu。

ContentMenu 位于 android.view 包下，它继承自 Menu。当 ContentMenu 注册于某个 View 对象上后，长按下该 View 对象时，系统会弹出上下文菜单。ContentMenu 的菜单项不支持快捷键，不附带图标，但是 ContentMenu 的标题可以指定图标。ContentMenu 中常用的方法如表 6-5 所示。

表 6-5 ContentMenu 中常用的方法及说明

方 法	参 数	描 述
onCreateContextMenu(ContextMenu menu, View v, ContextMenu. ContextMenuInfo menuInfo)	menu：创建的上下文菜单 v：上下文菜单依附的 View 对象 menuInfo：上下文菜单需要额外显示的信息	每次为 View 对象呼出上下文菜单时都调用
onContextItemSelected(MenuItem,item)	item：被选中的上下文菜单选项	当用户选择了上下文菜单选项后调用
onContextMenuClosed(Menu menu)	menu：被关闭的上下文菜单	当上下文菜单被关闭时调用
registerForContextMenu(View view)	view：要显示上下文菜单的 View 对象	为指定的 View 对象注册一个上下文菜单

在程序中创建一个上下文菜单需要如下步骤。

(1) 第一步，重写 Activity 的 onCreateContextMenu()方法，调用 Menu 的 add()方法添加菜单项 MenuItem。

(2) 第二步，重写 Activity 的 onContextItemSelected()方法，响应菜单单击事件。

(3) 第三步，调用 registerForContextMenu()方法，为视图 View 对象注册上下文菜单。

使用 registerForContextMenu()方法为 View 对象注册上下文菜单用法很简单，例如，要为一个资源 ID 名为“et01”的 EditText 控件注册上下文菜单，那么在类的 onCreate()方法中定义：

```
this. registerForContextMenu(findViewById(R. id. et01));
```

为 View 对象注册了上下文菜单后，Android 系统会自动地为指定的 View 对象添加一个 View. OnCreateContextMenuListener 监听器。这样，当长按该 View 时就会弹出上下文菜单。

6.2　对话框

对话框(Dialog)是 Activity 运行时弹出的小窗口。例如，当用户要删除一个联系人时，会弹出一个对话框，让用户确认是否真地要删除。当显示对话框时，当前的 Activity 失去焦点进入可见状态，而由对话框负责所有的人机交互。使用对话框的主要目的是提示消息，或弹出一个与程序主进程直接相关的小程序。

6.2.1　对话框简介

Android 系统主要提供 4 类对话框：提示对话框(AlertDialog)、对话框进度(ProgressDialog)、日期选择对话框(DatePickerDialog)和时间选择对话框(TimePickerDialog)。它们都位于 android. app 包下。

1. 常见的对话框

1) AlertDialog

AlertDialog 类定义在 android. app. AlertDialog 类中，它可以包含一些文本、单选按钮或复选框以及 0～4 个一般的按钮，是最常用的对话框，能够满足常见的对话框用户界面的需求。提示对话框 AlertDialog 可以进一步地细分为普通对话框、列表对话框、单选按钮对话框和复选对话框。

2) ProgressDialog

ProgressDialog 继承自 AlertDialog，可以显示进度条或者进度轮，并且在对话框中可以添加按钮。

3) DatePickerDialog

DatePickerDialog 对话框可以显示并允许用户修改日期。

4) TimePickerDialog

TimePickerDialog 对话框可以显示并允许用户修改时间。

2. 对话框的创建与使用

对话框是作为 Activity 的一部分被创建和显示的，在程序中通过重写一些回调方法来创建、保存、回复对话框。如果需要自定义对话框的外观等样式，可以继承 Dialog 或其子类并定义自己的布局。Dialog 常用的方法如表 6-6 所示。

表 6-6 Dialog 常用的方法及说明

方 法	参 数	描 述
onCreateDialog(int id)	id：代表该对话框的 ID	创建对话框，只在首次显示对话框时调用
showDialog(int id)	id：同上	显示指定 ID 的对话框
onPrepareDialog(int id, Dialog)	menu：被关闭的上下文菜单	每次显示对话框前调用
dismissDialog(int id)	id：代表该对话框的 ID	关闭指定 ID 的对话框
removeDialog(int id)	id：代表该对话框的 ID	关闭指定 ID 的对话框并彻底释放之

Android 系统通过 onCreateDialog()回调方法来完成对话框的创建，当创建好对话框后，在 Activity 的最后返回这个对象。注意，该方法只在对话框第一次被显示时执行，用于创建一个对话框实例，之后将不再重复创建该实例。

当要显示一个对话框时，调用 showDialog()方法并传递一个唯一标识这个对话框的整数 ID。如果在 Activity 中重写了 onPrepareDialog()方法，那么系统在调用 showDialog()方法之前会自动调用 onPrepareDialog()方法。在 onPrepareDialog()方法内可以对指定的对话框属性作设置，如果不重写该方法，那么每次显示的对话框将会是最初创建的那个。

当准备关闭对话框时，可以对 Activity 调用 dismissDialog()方法，或可以通过对该对话框调用 dismiss()来消除它。注意，用这种方式关闭的对话框不会彻底消失，在 Android 的后台会保留其状态，如果需要彻底清除对话框，需要调用 removeDialog()方法，将删除任何内部对象对其的引用而且如果这个对话框正在显示，它将被消除。

6.2.2 对话框案例

【案例 6.3】 在一个 Activity 里，通过一组按钮，分别打开几种常见的对话框。

【说明】 在本例中，所有按钮、对话框中显示的文字全部来自 strings. xml 和 array. xml 资源文件。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Activity_Dialog 的 Android 项目。其应用程序名为 Dialogs，包名为 cn. com. sgmsc. dialog，Activity 组件名为 DialogActivity。

(2) 准备图片。将图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 准备字符串资源。编写 res/values 目录下的 strings. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "hello">Hello World, DialogActivity!</string>
    <string name = "app_name">Dialogs</string>
    <string name = "btn1">显示普通对话框</string>  <!-- 声明名为 btn 的字符串资源 -->
    <string name = "title1">普通对话框</string>  <!-- 声明名为 title 的字符串资源 -->
    <string name = "dialog_msg1">这是普通对话框中的内容!!!</string>
    <string name = "btn2">显示列表对话框</string>
    <string name = "title2">列表对话框</string>
    <string name = "dialog_msg2">在列表中,你喜欢哪种运动?</string>
    <string name = "btn3">显示单选按钮对话框</string>
    <string name = "title3">单选按钮对话框</string>
```

```
    <string name = "dialog_msg3">单选,你喜欢哪种运动?</string>
    <string name = "btn4">显示复选框对话框</string>
    <string name = "title4">复选框对话框</string>
    <string name = "dialog_msg4">可多选,你喜欢哪种运动?</string>
    <string name = "btnok">显示进度对话框</string>
    <string name = "titleok">完成进度</string>
    <string name = "dialog_msgok">请稍等...</string>
    <string name = "date">显示日期选择对话框</string>
    <string name = "titledt">日期选择对话框</string>
    <string name = "dialog_msgdt">这是日期选择对话框中的内容!!!</string>
    <string name = "time">显示时间选择对话框</string>
    <string name = "titletm">时间选择对话框</string>
    <string name = "dialog_msgtm">这是时间选择对话框中的内容!!!</string>

    <string name = "ok">确定</string>
    <string name = "cancel">取消</string>
</resources>
```

(4) 创建数组资源。创建并编写 res/values 目录下的数组描述文件 array.xml,代码如下所示。

```
<?xml version = "1.0" encoding = "utf-8"?>
<resources>
  <string-array name = "msa">           <!-- 声明一个字符串数组 -->
     <item>游泳</item>                        <!-- 向数组中加入元素 -->
     <item>快走</item>                        <!-- 向数组中加入元素 -->
     <item>跑步</item>                        <!-- 向数组中加入元素 -->
     </string-array>
</resources>
```

(5) 设计布局。编写 res/layout 目录下的 main.xml 文件,代码如下所示。

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:paddingLeft = "20dip"
    android:paddingRight = "20dip"
    android:background = "@drawable/bg"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal"
    >                                         <!-- 声明一个线性布局 -->

    <EditText
        android:text = ""
        android:id = "@ + id/EditText01"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:editable = "false"
        android:cursorVisible = "false"
        />                                    <!-- 声明一个 EditText 控件 -->
    <Button
        android:text = "@string/btn1"
        android:id = "@ + id/Button01"
        android:layout_width = "180dip"
        android:layout_height = "wrap_content"
```

```
25            />                              <!-- 声明一个 Button1 控件 -->
26        <Button
27            android:text="@string/btn2"
28            android:id="@+id/Button02"
29            android:layout_width="180dip"
30            android:layout_height="wrap_content"
31            />                              <!-- 声明一个 Button2 控件 -->
32        <Button
33            android:text="@string/btn3"
34            android:id="@+id/Button03"
35            android:layout_width="180dip"
36            android:layout_height="wrap_content"
37            />                              <!-- 声明一个 Button3 控件 -->
38        <Button
39            android:text="@string/btn4"
40            android:id="@+id/Button04"
41            android:layout_width="180dip"
42            android:layout_height="wrap_content"
43            />                              <!-- 声明一个 Button4 控件 -->
44        <Button
45          android:text="@string/date"
46                android:id="@+id/Buttondt"
47                android:layout_width="180dip"
48                android:layout_height="wrap_content"
49                />                          <!-- 声明一个 Buttondt 控件 -->
50        <Button
51          android:text="@string/time"
52          android:id="@+id/Buttontm"
53          android:layout_width="180dip"
54          android:layout_height="wrap_content"
55          />                                <!-- 声明一个 Buttontm 控件 -->
56        <Button
57            android:text="@string/btnok"
58            android:id="@+id/Buttonok"
59            android:layout_width="180dip"
60            android:layout_height="wrap_content"
61            />                              <!-- 声明一个 Buttonok 控件 -->
62
63 </LinearLayout>
```

从第 20～61 行声明了 7 个按钮，每个按钮调用一个对话框。

（6）开发逻辑代码。打开 src/cn.com.sgmsc.dialog 包下的 DialogActivity.java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.dialog;                   //声明包语句

import java.util.Calendar;                     //引入相关类

import android.app.Activity;                   //引入相关类
import android.app.AlertDialog;                //引入相关类
import android.app.Dialog;                     //引入相关类
import android.app.AlertDialog.Builder;        //引入相关类
import android.app.ProgressDialog;             //引入相关类
import android.app.DatePickerDialog;           //引入相关类
import android.app.TimePickerDialog;           //引入相关类
```

```
12 import android.content.DialogInterface;          //引入相关类
13 import android.content.DialogInterface.OnClickListener;          //引入相关类
14 import android.os.Handler;                       //引入相关类
15 import android.os.Message;                       //引入相关类
16 import android.os.Bundle;                        //引入相关类
17 import android.view.View;                        //引入相关类
18 import android.widget.Button;                    //引入相关类
19 import android.widget.EditText;                  //引入相关类
20 import android.widget.DatePicker;                //引入相关类
21 import android.widget.TimePicker;                //引入相关类
22
23 public class DialogActivity extends Activity {
24  final int COMMON_DIALOG = 1;                    //普通对话框 id
25  final int LIST_DIALOG = 2;                      //声明列表对话框的 id
26  final int LIST_DIALOG_SINGLE = 3;               //记录单选列表对话框的 id
27  final int LIST_DIALOG_MULTIPLE = 4;             //记录复选按钮对话框的 id
28  boolean[] mulFlags = new boolean[]{true,false,true};          //初始复选情况
29  String[] items = null;                          //选项数组
30
31  final int PROGRESS_DIALOG = 5;                  //声明进度对话框 id
32  final int INCREASE = 6;                         //Handler 消息类型
33  ProgressDialog pd;
34  Handler myHandler;                              //Handler 对象引用
35
36  final int DATE_DIALOG = 6;                      //日期选择对话框 id
37  final int TIME_DIALOG = 7;                      //时间选择对话框 id
38  Calendar c = null;                              //声明一个日历对象
39
40    @Override
41    public void onCreate(Bundle savedInstanceState) {
42        super.onCreate(savedInstanceState);
43        setContentView(R.layout.main);            //设置当前屏幕
44        //打开普通对话框的按钮
45        Button btn1 = (Button) this.findViewById(R.id.Button01);          //获得 btn1 对象
46        btn1.setOnClickListener(new View.OnClickListener(){
                                                    //为 btn1 设置 OnClickListener 监听器
47            @Override
48            public void onClick(View v) {         //重写 onClick()方法
49                showDialog(COMMON_DIALOG);        //显示普通对话框
50            }
51      });
52
53     ……  //这里省略了若干个打开对话框按钮的代码段,这些代码与第 45～51 行相似
54
55        //打开进度对话框的按钮
56        Button bok = (Button)this.findViewById(R.id.Buttonok);          //获得 bok 对象
57        bok.setOnClickListener(                   //设置 OnClickListener 监听器
58            new View.OnClickListener(){
59               @Override
60               public void onClick(View v) {      //重写 onClick()方法
61                    showDialog(PROGRESS_DIALOG);          //显示进度对话框
62               }
63            }
64        );
65        myHandler = new Handler(){                //创建 Handler 对象
```

```
66          @Override
67          public void handleMessage(Message msg){
68              switch(msg.what){
69                  case INCREASE:
70                    pd.incrementProgressBy(1);  //进度每次加 1
71                        if(pd.getProgress()>=100){                  //判断是否结束进度
72                              pd.dismiss();     //如果进度条走完则关闭窗口
73                          }
74                  break;
75              }
76              super.handleMessage(msg);
77          }
78      };
79
80    }
81
82  @Override
83  protected Dialog onCreateDialog(int id) {         //重写 onCreateDialog()方法
84  ……//省略了该方法的代码段,相应代码将在后面给出解析
85  }
86
87    @Override
88    public void onPrepareDialog(int id, Dialog dialog){
                                        //每次弹出对话框时更新对话框内容的方法
89     super.onPrepareDialog(id, dialog);
90        switch(id){
91            case  PROGRESS_DIALOG:
92                pd.incrementProgressBy(-pd.getProgress());//对话框进度清零
93                new Thread(){                    //创建一个线程
94                    public void run(){
95                        while(true){
96                              myHandler.sendEmptyMessage(INCREASE);  //发送 Handler 消息
97                              if(pd.getProgress()>=100){
98                                  break;
99                              }
100                              try{
101                                  Thread.sleep(40);                 //线程休眠
102                              }
103                              catch(Exception e){
104                                  e.printStackTrace();               //捕获并打印异常
105                        }
106                      }
107                }
108               }.start();                                         //启动线程
109               break;
110         }
111     }
112
113 }
```

① 第 24～38 行声明了 7 个对话框的 ID 常量和与对话框相关的数组,需要引用的对象。

② 第 45～51 行获取按钮对象 btn1,并添加按钮监听,在监听中重写 onClick()方法。第 49 行使用方法 showDialog(COMMON_DIALOG)显示普通的对话框。紧跟后面的若干行代码均类似于第 45～51 行的代码段,在此省略了,完整代码参见本书指定网址上的源代码。

③ 第 56～64 行是定义按钮 bok 和添加监听，当单击此按钮时显示进度对话框。

④ 第 65～78 行创建一个 Handler 对象。当进度对话框显示时，定义进度条的前进速度；第 72 行定义当进度条走完时退出对话框。

⑤ 第 88～111 行重写了 onPrepareDialog()方法，主要是对进度对话框的进度进行清零处理。其中创建了一个线程，定义每隔 0.04s 更新一次进度。这段代码涉及线程方法的知识，将在第 8 章作详细介绍。

这部分省略的代码是重写 onCreateDialog()方法，用于定义各类对话框。其中使用 switch-case 语句分别对每个对话框进行定义。具体代码解析如下。

```
@Override
protected Dialog onCreateDialog(int id) {          //重写 onCreateDialog()方法
    Dialog dialog = null;                          //声明一个 Dialog 对象用于返回
    Builder b = new AlertDialog.Builder(this);
    switch(id){                                    //对 id 进行判断
    case COMMON_DIALOG:
        b.setIcon(R.drawable.ic_header);           //设置对话框的图标
        b.setTitle(R.string.btn1);                 //设置对话框的标题
        b.setMessage(R.string.dialog_msg1);        //设置对话框的显示内容
        b.setPositiveButton(                       //添加按钮
            R.string.ok,
            new OnClickListener() {                //为按钮添加监听器
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    EditText et = (EditText)findViewById(R.id.EditText01);
                    et.setText(R.string.dialog_msg1);//设置 EditText 内容
                }
            });
        dialog = b.create();                       //生成 Dialog 对象
        break;
    case LIST_DIALOG:
        b.setIcon(R.drawable.ic_header);           //设置图标
        b.setTitle(R.string.title2);               //设置标题
        b.setItems(                                //设置列表中的各个属性
                R.array.msa,                       //字符串数组
                new DialogInterface.OnClickListener() {
                                                   //为列表设置 OnClickListener 监听器
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        EditText et = (EditText)findViewById(R.id.EditText01);
                        et.setText("您选择了: "
                            + getResources().getStringArray(R.array.msa)[which]);
                    }
                });
        dialog = b.create();                       //生成 Dialog 对象
        break;
    case LIST_DIALOG_SINGLE:
        b.setIcon(R.drawable.ic_header);           //设置图标
        b.setTitle(R.string.title3);               //设置标题
        b.setSingleChoiceItems(                    //设置单选列表选项
                R.array.msa,
```

```
                    0,
                    new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                              EditText et = (EditText)findViewById(R.id.EditText01);
                              et.setText("您选择了: "
                                    + getResources().getStringArray(R.array.msa)[which]);
                        }
                    });
          b.setPositiveButton(                  //添加一个按钮
          R.string.ok,                          //按钮显示的文本
          new DialogInterface.OnClickListener() {
              @Override
              public void onClick(DialogInterface dialog, int which){}
          });
          dialog = b.create();                  //生成 Dialog 对象
          break;
     case LIST_DIALOG_MULTIPLE:
          b.setIcon(R.drawable.ic_header);      //设置图标
          b.setTitle(R.string.title4);          //设置标题
          b.setMultiChoiceItems(                //设置复选选项
                    R.array.msa,
                    mulFlags,                   //传入初始的选中状态
                    new DialogInterface.OnMultiChoiceClickListener() {
                        @Override
                          public void onClick(DialogInterface dialog, int which, boolean
                          isChecked) {
                              mulFlags[which] = isChecked;            //设置选中标志位
                              String result = "您选择了: ";
                              for(int i = 0;i < mulFlags.length;i++){
                                   if(mulFlags[i]){                 //如果该选项被选中
                                          result = result + items[i] + "、";
                                     }
                              }
                              EditText et = (EditText)findViewById(R.id.EditText01);
                              et.setText(result.substring(0,result.length() - 1));
                                                //设置 EditText 显示的内容
                        }
                    });
          b.setPositiveButton(                  //添加按钮
                    R.string.ok,
                    new DialogInterface.OnClickListener(){
                        @Override
                        public void onClick(DialogInterface dialog, int which) {}
          });
          dialog = b.create();                  //生成 Dialog 方法
          break;
    case DATE_DIALOG:                           //生成日期对话框的代码
        c = Calendar.getInstance();             //获取日期对象
        dialog = new DatePickerDialog(          //创建 DatePickerDialog 对象
           this,
           new DatePickerDialog.OnDateSetListener(){   //创建 OnDateSetListener 监听器
```

```
91                @Override
92                public void onDateSet(DatePicker dp, int year, int month,int dayOfMonth) {
93                    EditText et = (EditText)findViewById(R.id.EditText01);
94                    month = month + 1;
95                    et.setText("您选择了：" + year + "年" + month + "月" + dayOfMonth + "日");
96                }
97            },
98            c.get(Calendar.YEAR),               //传入年份
99            c.get(Calendar.MONTH),              //传入月份
100           c.get(Calendar.DAY_OF_MONTH)        //传入天数
101
102         );
103         break;
104     case TIME_DIALOG:                         //生成时间对话框的代码
105         c = Calendar.getInstance();           //获取日期对象
106         dialog = new TimePickerDialog(        //创建 TimePickerDialog 对象
107             this,
108             new TimePickerDialog.OnTimeSetListener(){ //创建 OnTimeSetListener 监听器
109               @Override
110               public void onTimeSet(TimePicker tp, int hourOfDay, int minute) {
111                   EditText et = (EditText)findViewById(R.id.EditText01);
112                   et.setText("您选择了：" + hourOfDay + "时" + minute + "分");
                                                  //设置 EditText 控件属性
113               }
114             },
115             c.get(Calendar.HOUR_OF_DAY),      //传入当前小时数
116             c.get(Calendar.MINUTE),           //传入当前分钟数
117             false
118         );
119         break;
120     case  PROGRESS_DIALOG:                    //创建进度对话框
121         pd = new ProgressDialog(this);        //创建进度对话框
122         pd.setMax(100);                       //设置最大值
123 //          pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);    //环形进度条
124         pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);     //水平进度条
125         pd.setTitle(R.string.titleok);        //设置标题
126         pd.setMessage("请稍等...");
127         pd.setCancelable(false);              //设置进度对话框不能用“回退”按钮关闭
128         dialog = pd;
129         break;
130     default:
131         break;
132     }
133     return dialog;                            //返回生成 Dialog 的对象
134 }
```

① 在 onCreateDialog()方法内使用 switch-case 语句定义各类对话框。

② 第 6～20 行定义一个普通对话框。分别使用 setIcon()、setTitle()和 setMessage()为对话框设置图标、标题和显示的内容；第 10～18 行为对话框添加“确定”按钮，其显示文本由第 11 行设置，第 12～18 行为该按钮添加 OnClickListener 监听，在其中定义当按钮被单击时，

在屏幕的 EditText 控件中显示来自字符串描述文件中的字符串内容；第 19 行生成该对话框。

③ 第 21～35 行定义一个列表对话框。使用 setItems 设置列表对话框的各属性，其中有两个参数，第一个参数是指定列表中的每一个 Item 的文本内容，第二个参数是为列表设置 OnClickListener 监听器。

④ 第 36～57 行定义一个单选列表对话框。使用 setSingleChoiceItems 设置单选列表选项，其中有三个参数，第一个参数是指定列表中的每一个 Item 的文本内容，第二个参数是默认被选中的项，这里取 0 表示第一项被选中，第三个参数是为列表设置 OnClickListener 监听器；第 50～55 行为对话框添加一个“确定”按钮并添加监听。

⑤ 第 58～85 行定义一个复选列表对话框。使用 setMultiChoiceItems 设置复选列表选项，其中有三个参数，第一个参数是指定列表中的每一个 Item 的文本内容，第二个参数是默认被选中的项，这里由一个数组提供选择项，第三个参数是为列表设置 OnClickListener 监听器；第 78～83 行为对话框添加一个“确定”按钮并添加监听。

⑥ 第 86～103 行定义一个日期对话框。使用 DatePickerDialog 创建一个日期对话框，其中添加了 OnDateSetListener 监听器。

⑦ 第 104～119 行定义一个时间对话框。使用 TimePickerDialog 创建一个时间对话框，其中添加了 OnTimeSetListener 监听器。

⑧ 第 120～131 行定义一个进度对话框，并设置进度对话框的属性。

⑨ 第 133 行返回生成 Dialog 的对象，结束 onCreateDialog()方法。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Activity_Dialog 项目。初始运行的显示效果如图 6-10 所示，当单击“显示普通对话框”按钮后屏幕显示如图 6-11 所示，当单击“显示列表对话框”按钮后屏幕显示如图 6-12 所示，当单击“显示单选按钮对话框”按钮后屏幕显示如图 6-13 所示，当单击“显示复选框对话框”按钮后屏幕显示如图 6-14 所示，当单击“显示日期选择对话框”按钮后屏幕显示如图 6-15 所示，当单击“显示时间选择对话框”按钮后屏幕显示如图 6-16 所示，当单击“显示进度对话框”按钮后屏幕显示如图 6-17 所示。



图 6-10 初始运行的显示效果



图 6-11 普通对话框



图 6-12 列表对话框



图 6-13 单选按钮对话框



图 6-14 复选框对话框



图 6-15 日期对话框



图 6-16 时间对话框



图 6-17 进度对话框

6.3 Android 应用案例

下面通过一个应用项目的中心框架设计来理解本章中所介绍的内容。

【案例 6.4】 完成“掌上微博”的日志查看与编辑功能。至少使用两个 Activity，第一个 Activity 以列表的方式列出以前写的日志内容，第二个 Activity 是创建或编辑日志。

【说明】 这两个 Activity 是实现日志功能的核心活动，它们有一些共同的信息要显示在界面上，如心情图标、日志标题和写日志时间等，有些变量和常量是两个 Activity 共同需要的。为避免重复声明常量，可以在项目的包内定义一个常量类，将多个 Activity 需要共用的常量定义在这个常量类中。本例将使用一个常量类。

在日志查看 Activity 中，设计一个选项菜单，用于新增或删除列表中的日志。如果新增日志，即跳转到日志编辑的 Activity 中，如果要删除某条日志，系统便弹出对话框以供用户确认删除操作。

在日志编辑的 Activity 中，只编写“返回”按钮的单击响应事件，对另外的按钮处理将在后续章节中完成。

【开发步骤及解析】

(1) Activity 的功能及界面设计。设计两个 Activity,一个是日志查看 Activity,用于显示日志信息列表,另一个是日志编辑 Activity,用于日志信息的输入和修改。

① 日志查看 Activity 界面。

✍以滚动方式,按条目显示每一条日志信息。具体包括:心情图标、日志标题、日志的建立或修改时间、“编辑”和“删除”按钮。

✍以选项菜单形式提供“添加”、“删除”日志菜单项。

✍单击“删除”日志菜单项时,有对话框确认。

✍单击“添加”日志菜单项时,进入日志编辑 Activity。

② 日志编辑 Activity 界面。

✍显示日志标题、建立日志的时间、心情图标、日志内容、“保存”和“返回”按钮。

✍新添加的日志开始不显示建立时间,待保存时将时间记入数据库。

✍心情图标可在预置的图标中选择。

✍每一条日志必须有标题和内容。

✍单击“返回”按钮,则返回到日志查看 Activity。

(2) 创建项目。在 Eclipse 中创建一个名为 ZSWB_Diary1 的 Android 项目。其应用程序名为 ZSWB,包名为 cn.com.sgmsc.ZSWB,Activity 组件名为 DiaryListActivity。

(3) 准备图片。这个项目需要准备一套图标,用于心情图标的显示,将制作的图片资源复制到本项目的 res/drawable-mdpi 目录中。

(4) 准备字符串资源。编写 res/values 目录下的 strings.xml 文件,代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "hello">Hello World, ZSWB!</string>
    <string name = "app_name">掌上微博</string>
    <string name = "ZSWB">掌上微博</string>

    <string name = "tvStatus">心情: </string>
    <string name = "tvTitle">标题: </string>
    <string name = "hintDiary">写点什么吧!</string>
    <string name = "btnBack">返回</string>
    <string name = "btnPublishMofify">发布日志</string>
    <string name = "btnDelete">删除</string>
    <string name = "btnEdit">编辑</string>
    <string name = "btnOk">确定</string>
    <string name = "btnAdd">添加</string>
    <string name = "btnCancel">取消</string>
    <string name = "dialog_message">确认删除此条日志吗?</string>

</resources>
```

(5) 创建颜色资源。创建并编写 res/values 目录下的颜色描述文件 colors.xml,参见本书源代码。

(6) 创建样式资源。创建并编写 res/values 目录下的样式描述文件 style.xml,参见本书源代码。

(7) 设计浏览日志布局。重命名 res/layout 目录下的 main.xml 文件为 diarylist.xml,并

编辑之。代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:background="@drawable/back"
     >
     <TextView
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="按 menu 键弹出添加菜单..."
        />
     <ScrollView
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:fillViewport="true"
       >
       <ListView
             android:id="@+id/listDiary"
             android:divider="@color/listDivider"
             android:dividerHeight="1px"
             android:layout_width="fill_parent"
             android:layout_height="fill_parent"
             android:choiceMode="singleChoice"
             />
       </ScrollView>
</LinearLayout>
```

(8) 设计编辑日志布局。创建并编写 res/layout 目录下的其他布局文件,名为 diarydetail.xml。代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:background="@drawable/back"
  android:layout_width="fill_parent"
  android:layout_height="match_parent"
  >
  <LinearLayout
       android:orientation="horizontal"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       >                                       <!-- 标题部分 -->
       <TextView
             android:text="@string/tvTitle"
             style="@style/text"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             />
       <EditText
             android:id="@+id/etModifyTitle"
             android:layout_width="fill_parent"
```

```
                android:layout_height = "wrap_content"
                android:singleLine = "true"
                />
        </LinearLayout>

<LinearLayout
        android:orientation = "horizontal"
        android:paddingLeft = "3dip"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:gravity = "center_vertical"
        >
        <TextView
                android:id = "@ + id/dttm"
                style = "@style/tvDatetime"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                 />                                <!-- 日期时间部分 -->
        <LinearLayout
                android:orientation = "horizontal"
                android:layout_width = "match_parent"
                android:layout_height = "wrap_content"
                android:gravity = "right"
                >                                  <!-- 心情部分 -->
                 <TextView
                        style = "@style/text"
                        android:text = "@string/tvStatus"
                        android:layout_width = "wrap_content"
                        android:layout_height = "wrap_content"
                        android:layout_gravity = "center_vertical"
                        />
                <Spinner
                        android:id = "@ + id/spinner_face"
                                android:layout_width = "wrap_content"
                                android:layout_height = "wrap_content"
                        android:layout_gravity = "center_vertical"
                            android:drawSelectorOnTop = "false"
                        />
                </LinearLayout>
        </LinearLayout>
 <LinearLayout
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
           android:layout_height = "fill_parent"
           android:layout_weight = "1"
      >                                            <!-- 内容部分 -->
      <EditText
            android:id = "@ + id/etModifyDiary"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:scrollbars = "vertical"
            android:gravity = "top"
            android:hint = "@string/hintDiary"
            />
      </LinearLayout>
```

```
<LinearLayout
    android:orientation = "horizontal"
    android:gravity = "bottom"
    android:layout_gravity = "center"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    >                                           <!-- 按钮部分 -->
    <Button
        android:id = "@ + id/btnModifyDiary"
        style = "@style/button"
        android:text = "@string/btnPublishMofify"
        android:layout_width = "120px"
        android:layout_height = "wrap_content"
        />
    <Button
        android:id = "@ + id/btnModifyDiaryBack"
        style = "@style/button"
        android:text = "@string/btnBack"
        android:layout_width = "120px"
        android:layout_height = "wrap_content"
        />
    </LinearLayout>
</LinearLayout>
```

(9) 开发常量类代码。在 src/cn. com. sgmsc. ZSWB 包下创建并编写常量类代码文件 ConstantUtil. java，代码如下所示。

```
1  package cn. com. sgmsc. ZSWB;
2
3  public class ConstantUtil {
4   public static int HEAD_HEIGHT  =  48;
5   public static int HEAD_WIDTH  =  48;
6   protected static final int[ ] FACEIDS = {
7        R. drawable. f_smile, R. drawable. f_tongue, R. drawable. f_embarrassed, R. drawable. f_love,
8        R. drawable. f_cool, R. drawable. f_ninja, R. drawable. f_huh, R. drawable. f_wink,
9        R. drawable. f_crazy, R. drawable. f_angry, R. drawable. f_worried, R. drawable. f_sadcry,
10       R. drawable. f_rolleyes, R. drawable. f_ohhmy, R. drawable. f_wow
11  };
12
13 }
```

第 7～11 行定义一个数组，该数组是所有的心情表情图标的资源 id。

(10) 开发日志浏览代码。打开 src/cn. com. sgmsc. ZSWB 包下的 DiaryListActivity. java 文件，并编辑之。这个文件是对日志查看的 Activity 进行逻辑代码设计，其代码如下所示。

```
package cn. com. sgmsc. ZSWB;

import static cn. com. sgmsc. ZSWB. ConstantUtil. FACEIDS;
import static cn. com. sgmsc. ZSWB. ConstantUtil. HEAD_HEIGHT;
import static cn. com. sgmsc. ZSWB. ConstantUtil. HEAD_WIDTH;

import android. app. Activity;
import android. app. AlertDialog;
import android. app. Dialog;
import android. app. AlertDialog. Builder;
```

```
import android.content.Intent;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.graphics.Color;

import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;

public class DiaryListActivity extends Activity {
 final int MENU_ADD = Menu.FIRST;                       //声明菜单选行的 ID
 final int MENU_DELETE = Menu.FIRST + 1;                //声明菜单项的编号
 final int DIALOG_DELETE = 0;                           //确认删除对话框的 ID
 String [] diarysTitle;                                 //声明用于存放日志标题的数组
 String [] diarysDatetime;                              //声明用于存放日志日期的数组
 int [] diarysFace;                                     //声明用于存放日志心情的数组
 int [] diarysId;                                       //声明用于存放日志 id 的数组
 int pos = -1;                                          //ListView 对象中的当前项位置
 ListView lv;                                           //声明 ListView 对象
 BaseAdapter myAdapter = new BaseAdapter(){
     @Override
     public int getCount() {
          if(diarysTitle != null){                      //如果日志数组不为空
            return diarysTitle.length;
          }
          else {
            return 0;                                   //如果日志数组为空则返回 0
          }
     }
     @Override
     public Object getItem(int arg0) {return null;}
     @Override
     public long getItemId(int arg0) {return 0;}
     @Override
     public View getView(int position, View convertView, ViewGroup parent) {
          LinearLayout ll = new LinearLayout(DiaryListActivity.this);//创建线性布局
          ll.setOrientation(LinearLayout.HORIZONTAL);
          ll.setGravity(Gravity.CENTER_VERTICAL);

          //定义日志列表的每条目的内容,包括心情表情图标、标题、创建日期等信息
          ImageView iv = new ImageView(DiaryListActivity.this);     //创建 ImageView 对象
          iv.setScaleType(ImageView.ScaleType.FIT_CENTER);
                                                    //设置图片等比缩放在显示区域中央
          iv.setImageDrawable(getResources().getDrawable(FACEIDS[diarysFace[position]]));
                                                                   //设置心情图片
```

```
            iv.setLayoutParams(new LinearLayout.LayoutParams(HEAD_WIDTH, HEAD_HEIGHT));
            iv.setPadding(3, 0, 0, 0);          //设置左边界空白

            LinearLayout ll2 = new LinearLayout(DiaryListActivity.this); //创建子线性布局
            ll2.setOrientation(LinearLayout.VERTICAL);
            ll2.setLayoutParams(new LinearLayout.LayoutParams(166,LayoutParams.WRAP_CONTENT));
            ll2.setPadding(3, 0, 0, 0);         //设置边界空白
            TextView tvTitle = new TextView(DiaryListActivity.this);
                                                //创建用于显示日志标题的 TextView
            tvTitle.setText(diarysTitle[position]); //设置日志标题的内容
            tvTitle.setTextSize(18.0f);         //设置字体大小
            tvTitle.setTextColor(Color.BLUE);   //设置字体颜色

            TextView tvDate = new TextView(DiaryListActivity.this);
                                                //创建用于显示日志创建时间的 TextView
            tvDate.setTextSize(18.0f);          //设置字体大小
            tvDate.setTextAppearance(DiaryListActivity.this, R.style.content);
            tvDate.setText(diarysDatetime[position]);       //设置日志创建时间的内容
            ll2.addView(tvTitle);               //将显示标题的 TextView 添加到线性布局
            ll2.addView(tvDate);                //将显示创建时间的 TextView 添加到线性布局

            LinearLayout llButton = new LinearLayout(DiaryListActivity.this);
                                                //创建子线性布局,布局按钮
            llButton.setOrientation(LinearLayout.HORIZONTAL);
            llButton.setLayoutParams(new LinearLayout.LayoutParams
                      (LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
            llButton.setPadding(3, 0, 3, 0);    //setPadding 参数: (left, top, right, bottom)
            llButton.setGravity(Gravity.RIGHT);
            Button btnEditDiary = new Button(DiaryListActivity.this);    //创建"编辑"按钮
            btnEditDiary.setTextAppearance(DiaryListActivity.this, R.style.button);
            btnEditDiary.setLayoutParams(new LinearLayout.LayoutParams(
                              50, LayoutParams.WRAP_CONTENT));
            btnEditDiary.setText(R.string.btnEdit);
            btnEditDiary.setId(position);       //设置 Button 的 ID
            btnEditDiary.setOnClickListener(listenerToEdit);        //设置按钮的监听器
            Button btnDeleteDiary = new Button(DiaryListActivity.this); //创建"删除"按钮
            btnDeleteDiary.setTextAppearance(DiaryListActivity.this, R.style.button);
            btnDeleteDiary.setLayoutParams(new LinearLayout.LayoutParams(
                               50, LayoutParams.WRAP_CONTENT));
            btnDeleteDiary.setText(R.string.btnDelete);
            btnDeleteDiary.setId(position);     //设置 Button 的 ID
            btnDeleteDiary.setOnClickListener(listenerToDelete);        //设置按钮的监听器
            llButton.addView(btnEditDiary);
            llButton.addView(btnDeleteDiary);
            ll.addView(iv);                     //将子对象添加到父布局中
            ll.addView(ll2);
            ll.addView(llButton);

            return ll;
    }
};
View.OnClickListener listenerToEdit = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(DiaryListActivity.this,DiaryDetailActivity.class);
```

```
            startActivity(intent);
        }
};
View.OnClickListener listenerToDelete = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            showDialog(DIALOG_DELETE);              //显示确认删除对话框,并实施相应操作
        }
};

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.diarylist);

        lv = (ListView)findViewById(R.id.listDiary); //获得 ListView 对象的引用
        lv.setAdapter(myAdapter);
    }

    //创建选项菜单
    @Override
public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, MENU_ADD, 0, R.string.btnAdd).setIcon(R.drawable.add);
                                                    //添加"添加"菜单项
        menu.add(0, MENU_DELETE, 0, R.string.btnDelete).setIcon(R.drawable.delete);
                                                    //添加"删除"菜单项
        return true;
}
//定义菜单项被选中后的回调事件
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()){                   //判断按下的菜单选项
        case MENU_ADD:                              //按下了"添加"菜单
            Intent intent = new Intent(DiaryListActivity.this,DiaryDetailActivity.class);
            startActivity(intent);
            break;
        case MENU_DELETE:                           //按下了"删除"菜单
            pos = DiaryListActivity.this.lv.getSelectedItemPosition();
                                                    //取 ListView 当前项的 ID
            showDialog(DIALOG_DELETE);              //显示确认删除对话框,并实施相应操作
            break;
        }
        return true;
}

//创建对话框
@Override
protected Dialog onCreateDialog(int id) {
        Dialog dialog = null;
        switch(id){                                 //对对话框 ID 进行判断
        case DIALOG_DELETE:                         //创建删除确认对话框
            Builder b = new AlertDialog.Builder(this);
            b.setIcon(R.drawable.dialog_delete);    //设置对话框图标
            b.setTitle("提示");                     //设置对话框标题
            b.setMessage(R.string.dialog_message);  //设置对话框内容
```

```
168            b.setPositiveButton(R.string.btnOk,
169                      new OnClickListener() {   //按下对话框中的"确认"按钮
170                      @Override
171                      public void onClick(DialogInterface dialog, int which) {
172                          deleteDiary(diarysId[pos]);     //调用删除处理方法
173                      }
174                  });
175            b.setNegativeButton(R.string.btnCancel,
176                      new OnClickListener() {   //按下对话框中的"取消"按钮
177                      @Override
178                      public void onClick(DialogInterface dialog, int which) {}
179                  });
180            dialog = b.create();
181            break;
182        }
183        return dialog;
184 }
185
186 //方法：删除指定的日志记录
187 public void deleteDiary(int id){                        //id 为要删除记录的 id
188      //......定义删除日志代码
189 }
190
191 }
```

① 第 40～111 行定义了一个 BaseAdapter 类的对象 myAdapter，用于显示日志的列表，当没有写任何日志时这个列表是空的。尽管刚开始这里是空的，但是在这里也要先定义好列表条目的显示布局。第 61～65 行定义了图标的布局属性；第 67～81 行定义一个内嵌的纵向线性布局，在其中添加两个 TextView 控件，分别显示日志标题和日志创建日期；第 83～104 行定义一个内嵌的横向线性布局，其中添加两个按钮，第一个按钮 btnEditDiary 被单击时，将跳转到 DiaryDetailActivity，这部分由第 112～118 行代码实现；第二个按钮 btnDeleteDiary 被单击时，将调出一个对话框，这部分由第 119～124 行代码实现。

② 第 137～141 行定义选项菜单的创建方法。

③ 第 144～156 行定义菜单项被选中后的回调事件，当单击"添加"菜单项时，跳转到 DiaryDetailActivity 中，当单击"删除"菜单项时，调出一个对话框。

④ 第 160～184 行定义创建对话框方法，这个对话框是在"删除"菜单项时被调出。其中，第 168～174 行定义"确定"按钮的 onClick 事件，它将执行删除指定的日志项内容；第 175～179 行定义"取消"按钮的 onClick 事件。

(11) 开发日志编辑代码。创建并编写 src/cn.com.sgmsc.ZSWB 包下的 DiaryDetailActivity.java 文件，该文件是对日志编辑的 Activity 进行逻辑代码设计，其代码如下所示。

```
package cn.com.sgmsc.ZSWB;

import static cn.com.sgmsc.ZSWB.ConstantUtil.FACEIDS;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Spinner;
import android.widget.BaseAdapter;
import android.widget.AdapterView.OnItemSelectedListener;

public class DiaryDetailActivity extends Activity{
 EditText etModifyTitle = null;                 //显示日志标题的 EditText
 EditText etModifyContent = null;               //显示日志内容的 EditText
 TextView tvdttm = null;                        //显示日志创建时间的 TextView
 Spinner facesp = null;                         //显示心情的 Spinner

 @Override
 protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.diarydetail);

      etModifyTitle = (EditText)findViewById(R.id.etModifyTitle);   //获得标题的 EditText
      etModifyContent = (EditText)findViewById(R.id.etModifyDiary); //获得内容的 EditText
      tvdttm = (TextView)findViewById(R.id.dttm);     //获得日期时间的 TextView

      facesp = (Spinner)findViewById(R.id.spinner_face);            //获得心情的 Spinner
      BaseAdapter ba = new BaseAdapter(){            //为 Spinner 准备内容适配器
           @Override
           public int getCount() {return 15;}       //总共 15 个选项
           @Override
           public Object getItem(int arg0) { return null; }
           @Override
           public long getItemId(int arg0) { return 0; }
           @Override
           public View getView(int arg0, View arg1, ViewGroup arg2) {
                  //初始化 ImageView
                  ImageView   iv = new ImageView(DiaryDetailActivity.this);
                  iv.setImageDrawable(getResources().getDrawable(FACEIDS[arg0]));
                                                     //设置图片
                  return iv;
           }
        };
        facesp.setAdapter(ba);                       //为 Spinner 设置内容适配器
        facesp.setOnItemSelectedListener(            //设置选项选中的监听器
           new OnItemSelectedListener(){
           @Override
           public void onItemSelected(AdapterView<?> arg0, View arg1,
                        int arg2, long arg3) {
             //......定义选项被选中事件的处理方法
           }
           @Override
           public void onNothingSelected(AdapterView<?> arg0) {}
            }
        );

      Button btnModifyBack = (Button)findViewById(R.id.btnModifyDiaryBack);
                                                     //获得"返回"按钮
      btnModifyBack.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
```

```
67                Intent intent = new Intent(DiaryDetailActivity.this,DiaryListActivity.class);
68                startActivity(intent);
69            }
70        });
71    }
72
73 }
```

① 第 35～49 行定义 BaseAdapter 对象 ba，绑定心情表情图标数据。

② 第 51～61 行为 facesp 添加 OnItemSelectedListener 监听。

③ 第 64～70 行设置“返回”按钮的 OnClickListener 监听。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 KDWB_Diary1 项目。初始运行时，当按下手机(或模拟器)中的 menu 键时，屏幕显示如图 6-18 所示，当单击“删除”菜单项时显示界面如图 6-19 所示，当单击“添加”菜单项时进入日志编辑界面如图 6-20 所示，在此界面单击“心情”下拉列表框时下拉出心情图标列表如图 6-21 所示。



图 6-18　按下 menu 键显示选项菜单



图 6-19　单击“删除”菜单调出对话框



图 6-20　单击“添加”菜单进入编辑日志界面

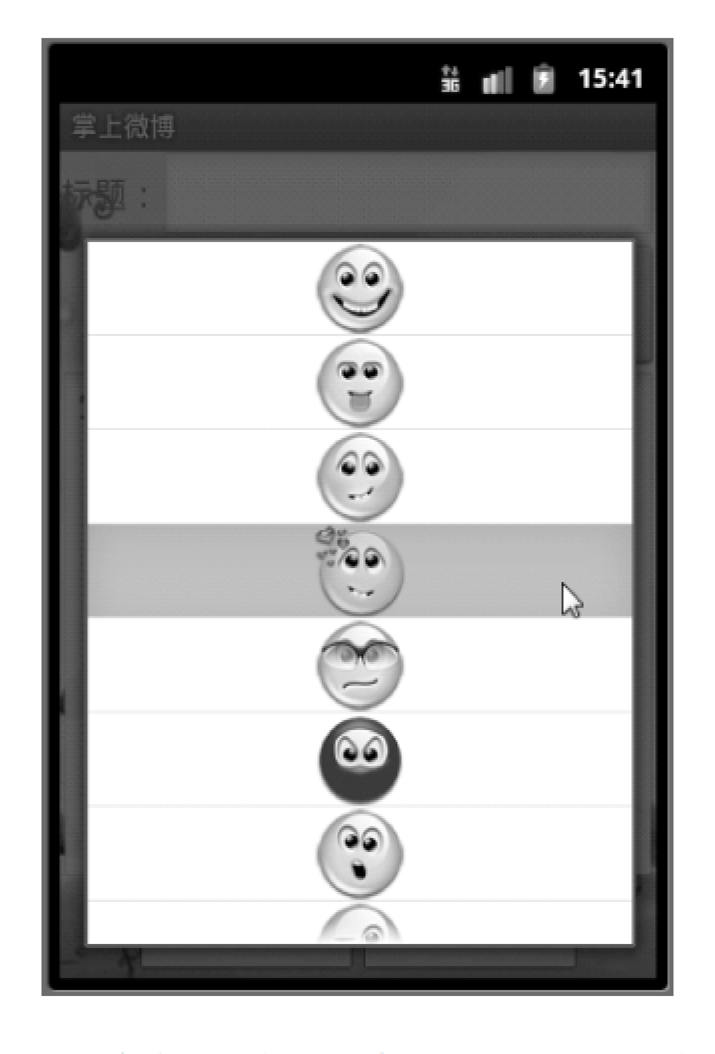


图 6-21　单击“心情”下拉列表，列出心情图标

小结

本章介绍了应用程序中常用的菜单和对话框的类型及其创建、设置和回调事件的处理。并举例说明了它们的使用。在创建并使用菜单和对话框时，有几个重要的方法及开发步骤必须理解掌握。

至本章，Android 系统提供的屏幕显示控件的介绍要告一段落了，通过学习读者已经具备了设计出各种各样用户界面的能力。接下来需要进阶学习 Android 应用程序的数据存储、消息传递、线程控制等知识，掌握 Android 应用的深层开发技术。

练习

设计“掌上微博”的相册管理模块用户界面。在“掌上微博”中用户可以定义多个相册，每个相册内包括多个照片、照片说明及拍摄日期等信息。要求至少使用两个 Activity，第一个 Activity 以列表的方式列出已有的相册标题，并使用选项菜单设置“新增”相册和“删除”相册菜单项；第二个 Activity 联合使用 ImageSwitcher 和 Gallery 控件显示指定相册中的照片，并添加“拍照”、“返回”按钮。

第7章 Android数据存储

程序本身就是数据的输入、处理和输出的过程，不管是操作系统还是应用程序都不可避免地要用到大量的数据。因此，数据存储是程序最基本的问题。在手机这种特殊设备里，也经常会进行存取数据处理，例如，存取通讯录、图片文件、音频文件和视频文件等数据的处理。

在 Android 系统中，所有应用程序的数据为该应用程序所私有，同时也提供了多个应用程序之间的数据通信标准方式。作为一种手机操作系统，Android 系统提供了以下几种数据存储和数据共享方式：Preference（配置）、File（文件）、SQLite 数据库、ContentProvider、SD Card 和网络。关于网络数据存储将在第 10 章中详细介绍，下面针对本地的几种存储方式进行介绍。

7.1 Preference 存储

Preference 提供了一种轻量级的数据存取方法，一般用于数据较少的配置信息的存储场合。例如，一些登录的用户名和密码，一些默认的应用项目问候词，程序关闭前界面的主要属性值等。使用 Preference 方式来存取数据，用到了 SharedPreferences 接口和 SharedPreferences 中的一个内部接口 SharedPreferences. Editor，这两个接口都在 android. content 包中。

SharedPreferences 类似于桌面操作系统中常用的 ini 文件，它以“键-值”对的方式将数据保存到一个内部的 XML 配置文件中，用来保存应用程序的一些属性设置。如果在 Android 的应用程序中使用了 SharedPreferences 保存数据，可以通过 ADT 的 DDMS 透视图来查看数据文件的位置。方法是在 Eclipse 的 DDMS 中，打开 File Explorer 标签页，展开到/data/data/<package name>/shared_prefs 下，可以看到相应的 XML 文件。以“掌上微博”为例，展开/data/data/cn. com. sgmsc. ZSWB/shared_prefs，可以看到“掌上微博”的配置文件 SPDATA_Files. xml，如图 7-1 所示。注意，只有在打开了 Android 的模拟器后，File Explorer 页内才有内容。

每个应用程序都有一个 SharedPreferences 对象。调用 Context. getSharedPreferences (String name，int mode) 方法获取 SharedPreferences 对象，其中第一个参数 name 为本组件的配置文件名；第二个参数 mode 是操作模式，操作模式有三种：MODE_PRIVATE（值为 0，应用程序私有，常用）、MODE_WORLD_READABLE（值为 1，其他程序可读）和 MODE_WORLD_WRITEABLE（值为 2，其他程序可写）。

SharedPreferences 提供了获得数据的方法，调用 SharedPreferences 的 edit() 方法返回

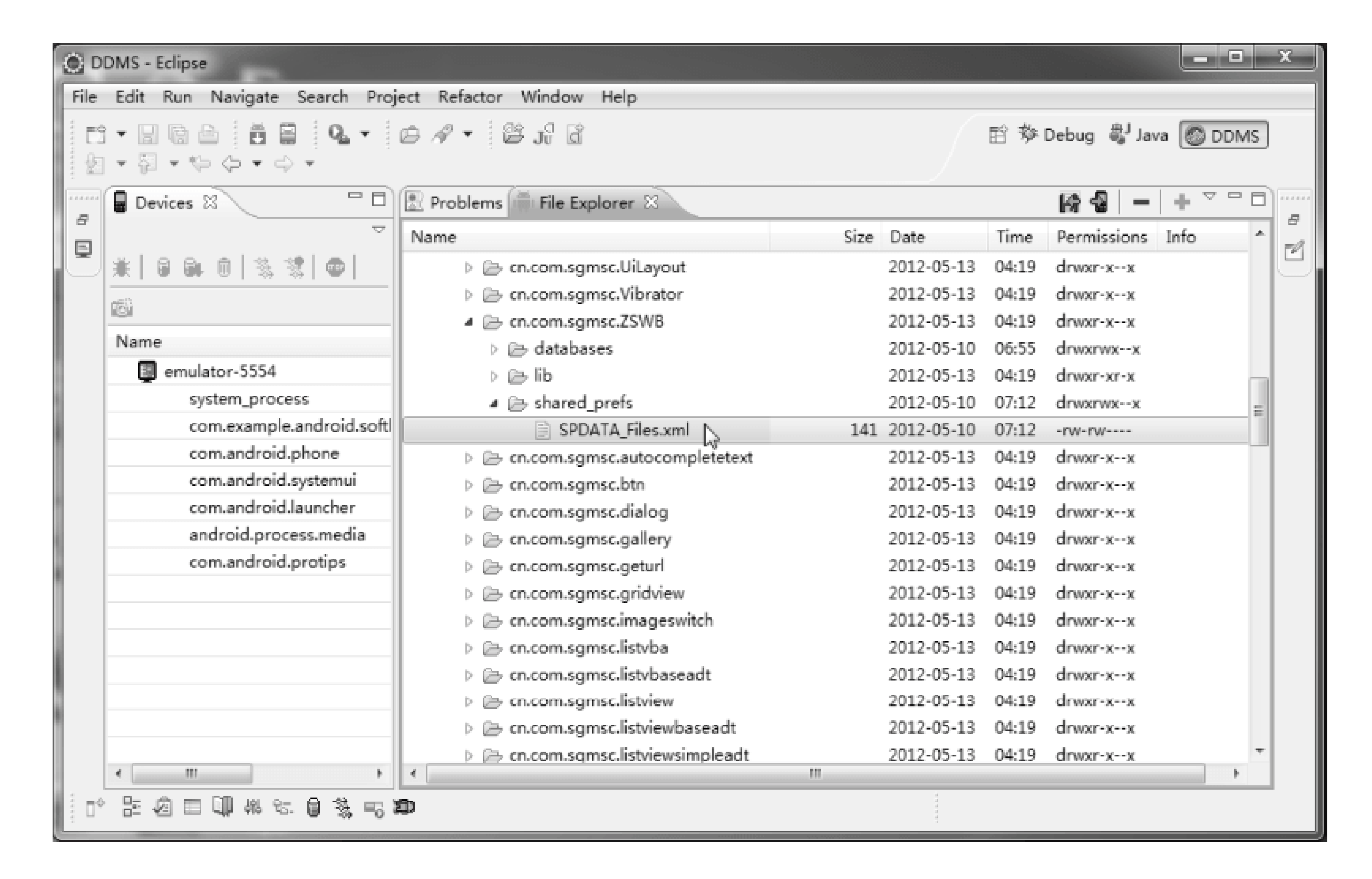


图 7-1　展开的/data/data/cn. com. sgmsc. ZSWB/shared_prefs 目录内容

SharedPreferences. Editor 内部接口，该接口中提供了保存数据的方法，这些常用的方法如表 7-1 和表 7-2 所示。

表 7-1　SharedPreferences 常用方法及说明

方　　法	描　　述
edit ()	返回 SharedPreferences 的内部接口 SharedPreferences. Editor
contains (String key)	判断是否包含该键值
getAll ()	返回所有配置信息的 Map
getBoolean(String key,boolean defValue)	获得一个 boolean 值
getFloat(String key,float defValue)	获得一个 float 值
getInt(String key,int defValue)	获得一个 int 值
getLong (String key,long defValue)	获得一个 long 值
getString (String key,String defValue)	获得一个 String 值

表 7-2　SharedPreferences. Editor 常用方法及说明

方　　法	描　　述
clear ()	清除所有值
commit()	保存
getAll ()	返回所有配置信息的 Map
putBoolean(String key,boolean value)	保存一个 boolean 值
putFloat (String key,float value)	保存一个 float 值
putInt(String key,int value)	保存一个 int 值
putLong (String key,long value)	保存一个 long 值
putString (String key,String value)	保存一个 String 值
remove (String key)	删除该键对应的值

SharedPreferences 对象的数据读取设计步骤为：获取 SharedPreferences 对象的键 key，再由 key 获取相应的键值，在这里，对于不同类型的键值有不同的 get()方法。SharedPreferences

对象的数据写入设计步骤为：通过 SharedPreferences 的 Editor 对象，设置键值，然后调用 commit()提交设置，写入 XML 文件。下面通过一个案例来说明 SharedPreferences 的具体用法。

【案例 7.1】 在“掌上微博”的用户登录界面中，增加一个复选框“记住我”。当勾选了该复选框后，系统的用户登录 Activity 将保存最近的一次用户登录信息。

【说明】 以第 4 章的案例 4.14 创建的项目 ZSWB_Login1 为基础，在界面布局中添加一个复选框，并在代码中增加如下生命周期方法的重定义：onStop()和 onDestroy()。

使用 SharedPreferences 对账号和密码的输入信息进行存取。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 ZSWB_Login2 的 Android 项目。其应用程序名为 ZSWB，包名为 cn.com.sgmsc.ZSWB，Activity 组件名为 Login2Activity。

(2) 准备图片。从项目 ZSWB_Login1 中复制图片资源到本项目的 res/drawable-mdpi 目录中。

(3) 准备其他资源。从项目 ZSWB_Login1 中复制 res/values 目录下的 colors.xml、strings.xml 和 styles.xml 文件到本项目的 res/values 目录中。

(4) 设计布局。从项目 ZSWB_Login1 中复制 res/layout 目录下的 login.xml 文件到本项目的 res/layout 目录中，并删除 main.xml 文件。修改 login.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout   ......   >                              <! -- 声明垂直分布的线性布局 -->
    <TextView   ......   >
    </TextView>
    <LinearLayout   ......      >
      <TextView   ......   />
      <EditText   ......   />
      </LinearLayout>
    <LinearLayout      ......   >
      <TextView   ......   />
      <EditText   ......   />
      </LinearLayout>

    <! -- 新增加的 LinearLayout -->
    <LinearLayout
       android:orientation = "horizontal"
       android:layout_gravity = "center_horizontal"
       android:layout_width = "wrap_content"
       android:layout_height = "wrap_content"
       >
       <CheckBox
           android:id = "@ + id/cbRemember"
             android:text = "@string/cbRemember"
             style = "@style/text"
             android:layout_width = "fill_parent"
             android:layout_height = "wrap_content"
             android:layout_gravity = "center_vertical"
             android:checked = "false"
             />
       </LinearLayout>

```

```
32      <LinearLayout  …        >              <!-- 声明与显示按钮的线性布局 -->
33          <Button   …    />
34          <Button   …    />
35          </LinearLayout>
36 </LinearLayout>
```

① 与案例 4.14 的布局文件相同代码在此省略，使用“……”代替。

② 第 22～28 行是本案例增加的内容。

(5) 开发逻辑代码。打开 src/cn.com.sgmsc.ZSWB 包下的 Login2Activity.java 文件，并编辑之，代码如下所示。

```
package cn.com.sgmsc.ZSWB;

import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.EditText;

public class Login2Activity extends Activity {

    public static final String SP_INFOS = "SPDATA_Files";
    public static final String USERID = "UserID";
    public static final String PASSWORD = "PassWord";
    private static EditText etUid;                       //接收用户 id 组件
    private static EditText etPwd;                       //接收用户密码组件
    private static CheckBox cb;                          //“记住我”复选框组件
    private static String uidstr;                        //用户账号
    private static String pwdstr;                        //用户密码

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.login);
      etUid = (EditText) findViewById(R.id.etUid);       //获得账号 EditText
      etPwd = (EditText) findViewById(R.id.etPwd);       //获得密码 EditText
      cb = (CheckBox) findViewById(R.id.cbRemember);     //获得 CheckBox 对象
      checkIfRemember();                  //从 SharedPreferences 中读取用户的账号和密码
   }
    @Override
    protected void onStop(){
      super.onStop();
      if(cb.isChecked()){
           uidstr = etUid.getText().toString().trim(); //获得输入的账号
           pwdstr = etPwd.getText().toString().trim(); //获得输入的密码
           rememberMe(uidstr,pwdstr);           //将用户的账号与密码存入 SharedPreferences
      }
    }
    @Override
    protected void onDestroy() {
     super.onDestroy();
    }
```

```
42
43      //方法：从 SharedPreferences 中读取用户的账号和密码
44      public void checkIfRemember(){
45        SharedPreferences sp = getSharedPreferences(SP_INFOS,MODE_PRIVATE);
46                                                              //获得 Preferences
47        uidstr = sp.getString(USERID, null);                  //取 Preferences 中的账号
48        pwdstr = sp.getString(PASSWORD, null);                //取 Preferences 中的密码
49        if(uidstr != null && pwdstr!= null){
50             etUid.setText(uidstr);                           //给 EditText 控件赋账号
51             etPwd.setText(pwdstr);                           //给 EditText 控件赋密码
52             cb.setChecked(true);
53        }
54      }
55
56      //方法：将用户的 id 和密码存入 SharedPreferences
57      public void rememberMe(String uid,String pwd){
58        SharedPreferences sp = getSharedPreferences(SP_INFOS,MODE_PRIVATE);
59        //获得 Preferences
60        SharedPreferences.Editor editor = sp.edit();          //获得 Editor
61        editor.putString(USERID, uid);                        //将用户账号存入 Preferences
62        editor.putString(PASSWORD, pwd);                      //将密码存入 Preferences
63        editor.commit();
64      }
65
66  }
```

① 第 11～13 行定义了三个字符串常量，分别代表配置文件名、账号的键名、密码的键名。

② 第 21～28 行重写 onCreate()方法。其中第 27 行的 checkIfRemember()是自定义方法，它将完成从 SharedPreferences 中读取用户的账号和密码。

③ 第 44～54 行是定义 checkIfRemember()方法。其中，第 45 行是获取 SharedPreferences 对象 sp；第 47、48 行分别取得账号值和密码值；第 51、52 行分别向账号和密码的 EditText 控件中传入取得的值，并将“记住我”复选框设置为选中状态。

④ 第 30～37 行重写 onStop()方法。其中第 35 行的 rememberMe(uidstr，pwdstr)是自定义方法，它将完成把用户的账号与密码存入到 SharedPreferences 中。

⑤ 第 57～64 行是定义 rememberMe(uidstr，pwdstr)方法。其中，第 60 行使用 edit()方法获得一个 SharedPreferences 的 Editor 对象 editor；第 61、62 行是将用户的账号值和密码值写到 editor 对象中；第 63 行将 editor 对象保存到 SharedPreferences 中。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行项目 KDWB_Login2。首次运行时因为没有输入任何信息，所以账号和密码都是空的，屏幕显示如图 7-2 所示。当在“账号”编辑框和“密码”编辑框中输入了相关信息，并勾选了“记住我”复选框后，退出该项目，然后再次进入 KDWB_Login2 时可以看到上次输入的账号和密码自动显示在编辑框内，屏幕显示如图 7-3 所示。

使用 SharedPreferences 的存储方式非常方便，但是它只适合存储比较简单且内容少的数据，如果需要存储更多的数据，就必须使用其他的存储方式了。



图 7-2 首次运行本项目的显示效果



图 7-3 选中“记住我”，再次运行的显示效果

7.2 文件存储

可以将一些数据以文件的形式保存在设备中，例如，一些文本文件、PDF 文件、图片文件、音频文件和视频文件等。

使用文件存储，Java 的 I/O 包中的常用类也可以正常工作在 Android 平台上，例如：Java. io. BufferedReader，Java. io. BufferedWriter，Java. io. FileReader，Java. io. FileWriter，Java. io. FileOutputStream，Java. io. FileInputStream，Java. io. OutputStream，Java. io. InputStream，Java. io. OutputStreamWriter，Java. io. InputStreamReader，Java. io. PrintStream，Java. io. PrintWriter，Java. io. FileNotFoundException 和 Java. io. IOException 等。

在 Android 中，可以通过 openFileOutput()方法打开一个指定的文件，通过 load()方法来获取文件中的数据，通过 deleteFile()方法来删除一个指定的文件，通过 openFileInput()方法写入文件，等等。例如使用文件代码段：

```
String FILE_NAME = "infofile.txt";                                          //获取要操作文件的文件名
FileOutputStream fos = openFileOutput(FILE_NAME,Context.NODE_PRIVATE);      //初始化一个文件流
FileInputStream fis = openFileInput(FILE_NAME);                             //创建写入文件流
```

注意，在使用上述文件流读写方法时，只能对当前应用程序所在目录下的文件进行操作。即指定的文件名中不能包含路径。如果调用 FileOutputStream 时指定的文件不存在，Android 系统会自动创建它。在默认情况下，写入操作是覆盖式的写入，如果需要把新的数据写入到文件中，而不覆盖原文件内容，则要将 openFileOutput()方法的第二个参数设置为 Context. MODE_APPEND。

在 Android 中对文件的存储操作与传统的 Java 对文件实现 I/O 的程序类似，在此不作赘述。

7.3 SQLite 数据库

Android 平台中内嵌一个轻量级的、功能强大的关系数据库 SQLite，用于完成各种复杂的数据处理，实现结构化的数据存储。

SQLite 数据库是一种关系数据库，支持多数的 SQL92 标准，最大支持数据库到 2TB。它没有服务进程，是一种嵌入到应用程序内部的数据库，所包含的数据库、表等所有数据都存放在一个单一的文件中。可以通过 ADT 的 DDMS 透视图来查看数据库文件的位置。即在 Eclipse 的 DDMS 中，打开 File Explorer 标签页，展开到/data/data/<package name>/databases 下，可以看到相应的数据库文件。例如，展开/data/data/cn.com.sgmsc.ZSWB/databases，可以看到“掌上微博”的数据库文件。在默认情况下，SQLite 数据库文件属于应用程序所私有，且数据库的名字是唯一的。

Android 提供了创建和使用 SQLite 数据库的 API，以及一些类与接口，下面分别介绍。

7.3.1 SQLite 数据库相关的类与接口

1. SQLiteDatabase 类

SQLiteDatabase 类位于 android.database.sqlite 包下，一个 SQLiteDatabase 对象代表一个数据库。在 Android 平台下，可以通过 SQLiteDatabase 类的静态方法创建或打开数据库，以及对数据库的记录进行增、删、改、查等操作。

2. SQLiteOpenHelper 类

SQLiteOpenHelper 类位于 android.database.sqlite 包下，它是一个辅助类，主要用来管理数据库的创建和版本。SQLiteOpenHelper 是一个抽象类，使用时通常需要创建子类继承它，并实现两个抽象方法 onCreate()和 onUpgrade()。

3. Cursor 接口

Cursor 位于 android.database 包下，它是 Android 的一个非常有用的游标接口，通过 Cursor 可以对数据库的查询结果集进行随机的读写访问。

4. ContentValues 类

ContentValues 类位于 android.content 包下，它存储一些键-值对，提供了数据库的列名、数据映射信息。ContentValues 的键是一个 String 类型的数据，对应数据库中的列名；而其相应的值是基本数据类型，对应数据库的数据，所以 ContentValues 对象代表了数据库的一行数据。

7.3.2 管理 SQLite 数据库相关的方法及编程

在 Android 中，通过上述类的相关方法来对 SQLite 数据库进行创建和管理。下面分别介绍常用的方法及其用法。

1. 相关方法

SQLiteDatabase 中常用的方法如表 7-3 所示。

表 7-3 SQLiteDatabase 的常用方法及说明

方　　法	参　　数	描　　述
openDatabase(String path,SQLiteDatabase.CursorFactory factory,int flags)	path：指定路径的数据库文件 factory：构造查询时返回的 Cursor 对象 flags：打开模式，模式参数包括： OPEN_READONLY(只读方式)； OPEN_READWRITE(可读可写)； CREATE_IF_NECESSARY(当数据库文件不存在时，创建该数据库)	打开或创建数据库
openOrCreateDatabase (String path, SQLiteDatabase.CursorFactory factory)	同上	相当于上面的方法中第三个参数取 CREATE_IF_NECESSARY 的情形
create (SQLiteDatabase. CursorFactory factory)	同上	创建一个内存数据库
insert(String table,String nullColumnHack,ContentValues values)	table：数据表名称 nullColumnHack：空列的默认值 values：封装了列名和列值的 Map	添加一条记录
delete(String table, String whereClause, String[] whereArgs)	table：数据表名称 whereClause：删除条件 whereArgs：删除条件值数组	删除一条记录
update (String table, ContentValues values, String whereClause, String [ ] whereArgs)	table：数据表名称 values：更新列 ContentValues 类型键-值对 whereClause：更新条件（即 where 子句） whereArgs：更新条件值数组	修改记录
query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)	table：数据表名称 columns：列名数组 selection：条件子句，相当于 where selectionArgs：条件子句，where 的值数组 groupBy：分组的列名 having：分组条件 orderBy：排序的列名	查询记录，其返回值为一个 Cursor 实例
execSQL(String sql)	sql：SQL 语句	执行一条 SQL 语句
close()		关闭数据库

Cursor 中常用的方法如表 7-4 所示。

表 7-4 Cursor 的常用方法及说明

方 法	描 述
getCount()	游标结果集的总记录条数
isFirst()	判断是否是第一条记录
isLast()	判断是否是最后一条记录
move(int offset)	移动到指定记录
moveToFirst()	移动到第一条记录
moveToLast()	移动到最后一条记录
moveToNext()	移动到下一条记录
moveToPrevious()	移动到上一条记录
getColumnIndexOrThrow(String columnName)	根据列名获得列索引
getInt(int columnIndex)	根据列索引获得 int 类型值
getString(int columnIndex)	根据列索引获得 String 类型值

SQLiteOpenHelper 中常用的方法如表 7-5 所示。

表 7-5 SQLiteOpenHelper 的常用方法及说明

方 法	描 述
onCreate (SQLiteDatabase db)	创建数据库时调用
onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion)	版本更新时调用
getReadableDatabase()	创建或打开一个只读数据库
getWritableDatabase ()	创建或打开一个读写数据库

在数据库第一次生成时会调用 onCreate()方法,一般在这个方法里写生成数据表的代码。当数据库需要升级时,Android 系统会自动调用 onUpgrade()方法,一般在这个方法里会删除数据表,并建立新的数据表,当然,还可以根据应用的需求,编写相关的代码完成其他的操作。

2. 相关操作

下面给出一些代码片断,说明如何使用这些方法对数据库、表进行操作。

1) 创建数据库

如果要在/data/data/cn. com. sgmsc. ZSWB/databases 目录下创建一个名为 pocketblog. db 的数据库,可使用方法:

```
openOrCreateDatabase("/data/data/cn.com. sgmsc.ZSWB/databases/pocketblog.db",null);
```

2) 创建数据表

在这里使用 execSQL()方法来创建一个数据表。首先,编写一条 SQL 语句存放到一个 String 变量中,然后再调用 execSQL()方法执行先前编写的 SQL 语句。例如,创建一个数据表 UserTb,其属性列为:uid(主键并且自动增加)、uname(用户名)、upsw(密码),可使用代码段:

```
String sql = "Create table UserTb(uid integer primary key autoincrement, uname text, upsw text)";
db.execSQL(sql);     //db 是指定的数据库对象
```

3) 插入数据

分别使用 execSQL()方法和 insert()方法来完成向数据表 UserTb 插入记录,以便读者更

好地理解 insert()方法的使用。代码段如下：

```
String sql = "insert into UserTb(uname,upsw) values('user01','123')";
db.execSQL(sql);     //执行 SQL 语句,向 db 的数据表 UserTb 中插入一条记录
```

或

```
ContentValues mycv = new ContentValues();
mycv.put("uname"," user01'");
mycv.put("upsw"," 123'");
db.insert("UserTb",null,mycv);                    //向 db 的数据表 UserTb 中插入一条记录
```

4）删除数据

和插入数据类似，使用两种方法来实现删除数据表 UserTb 中的记录。代码段如下：

```
String sql = "delete from UserTb where uid = 2";
db.execSQL(sql);     //执行 SQL 语句,在 db 的数据表 UserTb 中删除 uid 为 2 的记录
```

或

```
String whereClause = "uid = ?";
String[] whereArgs = {String.valueOf(2)};
db.delete("UserTb", whereClause, whereArgs);      //在 db 的数据表 UserTb 中删除 uid 为 2 的记录
```

这里，String. valueOf(2)是将数值 2 转换成字符串值“2”。String. valueOf()是将其他基本数据类型转换成 String 的静态方法，例如：String. valueOf(int i) 将 int 变量 i 转换成字符串，String. valueOf(boolean b) 将 boolean 变量 b 转换成字符串，String. valueOf(char c) 将 char 变量 c 转换成字符串，等等。

5）修改数据

和插入或删除数据类似，使用两种方法来实现修改数据表 UserTb 中的记录。代码段如下：

```
String sql = "update UserTb set upsw = 666 where uid = 1";
db.execSQL(sql);                 //执行 SQL 语句,在 db 的数据表 UserTb 中修改 uid 为 1 的记录
```

或

```
ContentValues newvalue = new ContentValues();
newvalue.put("upsw"," 666");
String whereClause = "uid = ?";
String[] whereArgs = {String.valueOf(1)};
db.update("UserTb",newvalue, whereClause, whereArgs);
                                                  //在 db 的数据表 UserTb 中修改 uid 为 1 的记录
```

6）查询数据

查询数据方法将返回一个 Cursor 对象，这里存放的是查询数据表的结果集。在应用编程中，往往要使用 Cursor 的方法，才能在让用户看到查询结果集中的内容。例如要查询 UserTb 中 uid 大于 5 的用户信息，可使用代码段：

```
Cursor qc = db.query("UserTb",null,"uid>?",new String[] { String.valueOf(5)},null,null,null);
if(qc.moveToFirst()){                            //判断游标是否为空
   for (int i = 0; i<qc. getCount(); i++){       //遍历游标
     qc.move(i);
```

```
        Int userid = qc.getInt(0);                  //获得用户 id
        String username = qc.getString(1);          //获得用户名
        String userpassword =  qc.getString(2);     //获得用户密码
        System.out.println(userid + ": " + username + "," + userpassword);
                                                    //在屏幕上显示用户 id,用户名和密码
        }
    }
}
```

以上主要是针对 SQLiteDatabase 类给出的对 SQLite 数据库进行操作的使用方法。在实际开发中,比较常见的是开发一个数据库辅助类来方便地对数据库进行操作。下来通过一个案例,来说明如何使用 SQLiteOpenHelper 辅助类来实现 SQLite 数据库应用。

7.3.3 SQLite 应用案例

【案例 7.2】 将“掌上微博”的日志信息存储到数据库中。

【说明】 在第 6 章的案例 6.4 中已经完成了日志模块的用户界面设计,并实现了日志列表与日志编辑两个 Activity 的跳转功能。本案例以项目 ZSWB_Diary1 为基础,继续完善数据保存功能。

本例使用 SQLiteDatabase、SQLiteOpenHelper、ContentValues、Cursort 来实现将日志信息保存到本地的 SQLite 数据库中的功能。SQLiteOpenHelper 是一个抽象类,使用时需要创建一个子类,以便在其 onCreate()方法中完成数据库表结构的定义。

【开发步骤及解析】

(1) 数据库设计。“掌上微博”项目中需要保存到数据库中的信息不少,在此只设计与日志相关的数据表。在该项目中,命名数据库名为 PocketBlog,命名日志数据表名为 DiarysTb。数据表 DiarysTb 的结构如表 7-6 所示。

表 7-6 DiarysTb 表结构

列名	类型	说　明	列名	类型	说　明
d_id	integer	日志 ID,主键	d_face	integer	日志心情索引
d_title	varchar	日志标题	d_datetime	datetime	写日志日期时间
d_content	text	日志内容	d_uno	varchr	日志所有者账号

(2) 创建项目。在 Eclipse 中创建一个名为 ZSWB_Diary2 的 Android 项目。其应用程序名为 ZSWB,包名为 cn.com.sgmsc.ZSWB,Activity 组件名为 DiaryListActivity。

(3) 准备图片。从项目 ZSWB_Diary1 中复制图片资源到本项目的 res/drawable-mdpi 目录中。

(4) 准备其他资源。从项目 ZSWB_Diary1 中复制 res/values 目录下 colors.xml,strings.xml 和 styles.xml 文件到本项目的 res/values 目录中。

(5) 设计布局。从项目 ZSWB_Diary1 中复制 res/layout 目录下 diarylist.xml 和 diarydetail.xml 文件到本项目的 res/layout 目录中,并删除 main.xml 文件。

(6) 开发常量类代码。从项目 ZSWB_Diary1 中复制 src/cn.com.sgmsc.ZSWB 包下 ContantUtil.java,DiaryListActivity.java 和 DiaryDetailActivity.java 文件到本项目的 src/cn.com.sgmsc.ZSWB 包中。

（7）开发创建数据库表代码。在 src/cn.com.sgmsc.ZSWB 包下创建并编写数据库辅助类子类代码文件 MyOpenHelper.java，代码如下所示。

```
1  package cn.com.sgmsc.ZSWB;                                    //声明包语句
2
3  import android.content.Context;
4  import android.database.sqlite.SQLiteDatabase;
5  import android.database.sqlite.SQLiteOpenHelper;
6  import android.database.sqlite.SQLiteDatabase.CursorFactory;
7
8  public class MyOpenHelper extends SQLiteOpenHelper{
9   public static final String DB_NAME = "PocketBlog";         //数据库文件名称
10  public static final String TABLE_NAME = "DiarysTb";        //表名
11  public static final String ID = "d_id";                    //ID
12  public static final String TITLE = "d_title";              //日志标题
13  public static final String CONTENT = "d_content";          //日志内容
14  public static final String FACE = "d_face";                //日志心情
15  public static final String DATETIME = "d_datetime";        //日志发布日期
16  public static final String USERNO = "d_uno";               //日志所属用户 ID
17
18  //调用父类构造器
19  public MyOpenHelper(Context context, String name, CursorFactory factory, int version) {
20        super(context, name, factory, version);
21  }
22  @Override                                                  //重写 onCreate()方法
23  public void onCreate(SQLiteDatabase db) {
24        db.execSQL("create table if not exists " + TABLE_NAME + " (" //调用 execSQL()方法创建表
25                  + ID + " integer primary key,"
26                  + TITLE + " varchar,"
27                  + CONTENT + " text,"
28                  + FACE + " integer,"
29                  + DATETIME + " datetime,"
30                  + USERNO + " varchar)");
31  }
32
33  @Override
34  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
35        //重写 onUpgrade()方法
36  }
37 }
```

① 第 9～16 行定义一些字符串常量，分别用来表示数据库名、数据表名以及表中的列名

② 第 23～31 重写 SQLiteOpenHelper 类的 onCreate()方法，在其中调用 SQLiteDatebase 的 execSQL()方法创建数据表，execSQL 内的参数值是一条完整的创建表的 SQL 语句。这个方法在数据库首次生成时自动被调用。

（8）开发浏览日志代码。修改 src/cn.com.sgmsc.ZSWB 包下的代码文件 DiaryListActivity.java。增加的代码或与原代码不同处使用粗体显示，代码如下所示。

```
package cn.com.sgmsc.ZSWB;

import static cn.com.sgmsc.ZSWB.MyOpenHelper.*;
import android.database.Cursor;
```

```
import android.database.sqlite.SQLiteDatabase;
//…… 省略部分引入相关类的代码

public class DiaryListActivity extends Activity {
 MyOpenHelper myHelper;                        //声明 MyOpenHelper 对象
 final int MENU_ADD = Menu.FIRST;              //声明菜单选行的 ID
 final int MENU_DELETE = Menu.FIRST + 1;       //声明菜单项的编号
 final int DIALOG_DELETE = 0;                  //确认删除对话框的 ID
 String [] diarysTitle;                        //声明用于存放日志标题的数组
 String [] diarysDatetime;                     //声明用于存放日志日期的数组
 int [] diarysFace;                            //声明用于存放日志心情的数组
 int [] diarysId;                              //声明用于存放日志 id 的数组
 int pos = - 1;                                //ListView 对象中的当前项位置
 ListView lv;                                  //声明 ListView 对象
 BaseAdapter myAdapter = new BaseAdapter(){
       //…… 省略构造 BaseAdapter 的部分代码
       }
 };
 View.OnClickListener listenerToEdit = new View.OnClickListener() {
       @Override
       public void onClick(View v) {
           pos = v.getId();                    //取 ListView 当前项的 ID
           Intent intent = new Intent(DiaryListActivity.this,DiaryDetailActivity.class);
           intent.putExtra("cmd", 0);          //0 代表查看或修改日志,1 代表添加日志
           intent.putExtra("id", diarysId[pos]);
           startActivity(intent);
       }
 };
 View.OnClickListener listenerToDelete = new View.OnClickListener() {
       @Override
       public void onClick(View v) {
                pos = v.getId();               //取 ListView 当前项的 ID
           showDialog(DIALOG_DELETE);          //显示确认删除对话框,并实施相应操作
       }
 };

@Override
public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.diarylist);

       myHelper = new MyOpenHelper(this, DB_NAME, null, 1);  //打开数据表库表

       lv = (ListView)findViewById(R.id.listDiary);          //获得 ListView 对象的引用
       lv.setAdapter(myAdapter);
 }

 @Override
 protected void onResume() {
       getBasicInfo(myHelper);                 //重新获取数据库信息
       myAdapter.notifyDataSetChanged();       //刷新 ListView
       super.onResume();
 }

     //方法: 获取数据表中所有记录部分列的数据内容
```

```
    public void getBasicInfo(MyOpenHelper helper){
      SQLiteDatabase db = helper.getWritableDatabase(); //获取数据库连接
      Cursor c = db.query(TABLE_NAME, new String[]{ID,TITLE,DATETIME,FACE},
                          null, null, null, null, ID);
      int idIndex = c.getColumnIndex(ID);
      int titleIndex = c.getColumnIndex(TITLE);          //获得标题列的列号
      int dtIndex = c.getColumnIndex(DATETIME);          //获得日期时间列的序号
      int faceIndex = c.getColumnIndex(FACE);            //获得心情列的索引号
      diarysId = new int[c.getCount()];                  //创建存放 id 的 int 数组对象
      diarysTitle = new String[c.getCount()];            //创建存放标题的 String 数组对象
      diarysDatetime = new String[c.getCount()];         //创建存放日期时间的数组对象
      diarysFace = new int[c.getCount()];                //创建存放心情 id 的数组对象
      int i = 0;                                         //声明一个计数器
      for(c.moveToFirst();!(c.isAfterLast());c.moveToNext()){
          diarysId[i] = c.getInt(idIndex);
          diarysTitle[i] = c.getString(titleIndex);      //将标题添加到 String 数组中
          diarysDatetime[i] = c.getString(dtIndex);      //将日期时间添加到 String 数组中
          diarysFace[i] = c.getInt(faceIndex);           //将心情 id 添加到 String 数组中
          i++;
      }
      c.close();                                         //关闭 Cursor 对象
      db.close();                                        //关闭 SQLiteDatabase 对象
    }

    //创建选项菜单
    @Override
  public boolean onCreateOptionsMenu(Menu menu) {
        //……
  }
  //定义菜单项被选中后的回调事件
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()){                        //判断按下的菜单选项
        case MENU_ADD:                                   //按下了“添加”菜单
            Intent intent = new Intent(DiaryListActivity.this,DiaryDetailActivity.class);
            intent.putExtra("cmd", 1);
            startActivity(intent);
            break;
        case MENU_DELETE:                                //按下了“删除”菜单
            pos = DiaryListActivity.this.lv.getSelectedItemPosition();
                                                         //取 ListView 当前项的 ID
            showDialog(DIALOG_DELETE);               //显示确认删除对话框,并实施相应操作
            break;
        }
        return true;
}

//创建对话框
@Override
protected Dialog onCreateDialog(int id) {
      //……
}

//方法: 删除指定的日志记录
public void deleteDiary(int id){                         //id 为要删除记录的 id
```

```
114         SQLiteDatabase db = myHelper.getWritableDatabase();     //获得数据库对象
115         try{
116         int count = db.delete(TABLE_NAME, ID + " = ?", new String[]{id + ""});
117         db.close();
118         if(count == 1){                                        //删除成功
119               getBasicInfo(myHelper);                          //重新获取数据库信息
120               myAdapter.notifyDataSetChanged();                //刷新 ListView
121         }
122          }catch(Exception e){
123                       e.printStackTrace();
124          }
125     }
126 }
```

① 第 3～5 行增加了一些引入类，其中有用户自定义的类 MyOpenHelper。

② 第 9 行增加了声明一个 MyOpenHelper 类对象 myHelper。

③ 修改了对 BaseAdapter 中两个按钮的监听对象的定义。在定义的监听对象 listenerToEdit 中，增加了第 26 行获取 ListView 当前条目的 ID，增加了第 28、29 行为 Intent 对象传入附加信息，其中 diarysId[pos]是取日志在数据表中的 ID 号；在定义的监听对象 listenerToDelete 中，增加了第 36 行，为删除表记录提供 ID 信息。

④ 在 onCreate()方法中增加了第 46 行使用 MyOpenHelper 打开数据库表，第一次调用时数据库表不存在，会自动创建指定名称的数据库和数据表。

⑤ 增加了第 52～57 行，定义 onResume()方法，当 Activity 与用户开始交互之前，通过 getBasicInfo(myHelper)获取数据表中的记录值；通过 myAdapter. notifyDataSetChanged()方法刷新屏幕中的 ListView 中的列表信息，notifyDataSetChanged 是通过一个外部的方法控制：如果适配器的内容改变时需要强制调用 getView()来刷新每个 Item 的内容。因为本例将对数据表中的信息做增、删、改等操作，相应地，屏幕显示也需要同步更新。

⑥ 第 60～82 行定义了 getBasicInfo()方法。其中，第 61 行创建并打开一个数据库对象 db；第 62 行对数据表的日志 ID、日志标题、日志日期、日志心情图标的索引进行查询；第 72～79 行遍历该查询结果游标，分别将数据表的上述列中的值赋予相应的数组中；第 80、81 行是关闭游标对象和数据库对象。记住要养成一个好习惯，当对数据库、数据表进行完操作之后要即时关闭之。

⑦ 增加了第 95 行，当单击了"添加"菜单项时，将一个状态附加信息传入到 Intent 对象中。

⑧ 第 113～125 行定义了 deleteDiary()方法。其中，第 116 行使用 delete()方法删除数据表中的指定 ID 的记录，并将返回删除操作执行的状态值，当其返回为 1 时表示删除成功，否则删除失败；第 118～124 行是对删除成功后所做的系列处理：重新获取数据表的数据集，更新屏幕的日志列表。

(9) 开发编辑日志代码。修改 src/cn. com. sgmsc. ZSWB 包下的代码文件 DiaryDetailActivity. java。增加的代码或与原代码不同处使用粗体显示，代码如下所示。

```
package cn.com.sgmsc.ZSWB;

import static cn.com.sgmsc.ZSWB.MyOpenHelper.*;
import android.content.ContentValues;
import android.text.format.DateFormat;
```

```
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
//…… 省略部分引入相关类的代码

public class DiaryDetailActivity extends Activity{
 EditText etModifyTitle = null;                  //显示日志标题的 EditText
   EditText etModifyContent = null;              //显示日志内容的 EditText
   TextView tvdttm = null;                       //显示日志创建时间的 TextView
   Spinner facesp = null;                        //显示心情的 Spinner
   int status = -1;                              //0 表示查看或修改日志,1 表示添加日志
   MyOpenHelper myHelper;                        //声明一个 MyOpenHelper 对象
   int id = -1;                                  //记录当前显示的日志 id
   String [] diaryInfo = null;                   //记录日志信息
   int faceselectedid = -1;                      //心情的 id 号
   DateFormat df = null;                         //显示日期时间的格式

   @Override
   protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.diarydetail);

        //…… 省略部分代码
        facesp.setOnItemSelectedListener(        //设置选项选中的监听器
              new OnItemSelectedListener(){
              @Override
              public void onItemSelected(AdapterView<?> arg0, View arg1,
                        int arg2, long arg3) {   //重写选项被选中事件的处理方法
              faceselectedid = arg2;
              }
              @Override
              public void onNothingSelected(AdapterView<?> arg0) {}
               }
             );

        myHelper = new MyOpenHelper(this, MyOpenHelper.DB_NAME, null, 1);
        Intent intent = getIntent();
        status = intent.getExtras().getInt("cmd");//读取命令类型

        switch(status){
        case 0:                                  //查看或修改日志的详细信息
            id = intent.getExtras().getInt("id"); //获得要显示的日志的 id
            SQLiteDatabase db = myHelper.getWritableDatabase();
            Cursor c = db.query(MyOpenHelper.TABLE_NAME,
                          new String[]{TITLE,CONTENT,FACE,DATETIME,USERNO},
                          ID + " = ?",
                          new String[]{id + ""}, null, null, null);
            if(c.getCount() == 0){               //没有查询到指定的日志
                                                 //显示没有找到指定的日志信息
            }
            else{                                //查询到了这条日志
                  c.moveToFirst();               //移动到第一条记录
                  etModifyTitle.setText(c.getString(0));
                  etModifyContent.setText(c.getString(1));
                  facesp.setSelection(c.getInt(2), true);       //设置指定的心情图标项
                  tvdttm.setText(c.getString(3));
```

```
                }
                c.close();
                db.close();
                break;
            case 1:                                          //新建日志的详细信息
                etModifyTitle.getEditableText().clear();     //清空 EditText 项
                etModifyContent.getEditableText().clear();
                facesp.setSelection(0, true);                //将心情设置为第一个图标项
                break;
            }

            Button btnModifyDiary = (Button)findViewById(R.id.btnModifyDiary);
                                                             //获得“发布日志”按钮
            btnModifyDiary.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    String modifyTitle = etModifyTitle.getEditableText().toString().trim();
                                                             //取出 EditText 中标题的内容
                    String modifyContent = etModifyContent.getEditableText().toString().trim();
                    if(modifyContent.equals("") || modifyTitle.equals("")){
                                                             //如果标题或内容为空则提示
                        //提示需要将标题或内容填写完整
                        return;
                    }
                    df = new android.text.format.DateFormat();//定义一个日期格式对象
                    tvdttm.setText(df.format("yyyy - MM - dd hh:mm:ss", new java.util.Date()));
                                                             //取当前的日期
                    switch(status){                          //判断当前的状态
                    case 0:                                  //编辑或查看已有日志时
                        updateDiary();                       //调用更新日志方法
                        break;
                    case 1:                                  //新建日志时
                        insertDiary();                       //调用插入日志方法
                        break;
                    }
                    //返回到日志列表 Activity
                    Intent intent = new Intent(DiaryDetailActivity.this,DiaryListActivity.class);
                    intent.putExtra("id", id);
                    startActivity(intent);
                }
            });
            Button btnModifyBack = (Button)findViewById(R.id.btnModifyDiaryBack);
                                                             //获得“返回”按钮
            btnModifyBack.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent = new Intent(DiaryDetailActivity.this,DiaryListActivity.class);
                    intent.putExtra("id", id);
                    startActivity(intent);
                }
            });
    }

//方法:更新某个日志
public void updateDiary(){
```

```
112        SQLiteDatabase db = myHelper.getWritableDatabase();                   //获得数据库对象
113        ContentValues values = new ContentValues();
114        values.put(TITLE, etModifyTitle.getText().toString());
115        values.put(CONTENT, etModifyContent.getText().toString());
116        values.put(FACE, faceselectedid);
117        values.put(DATETIME, tvdttm.getText().toString());
118        db.update(TABLE_NAME, values, ID + " = ?", new String[]{id + ""});  //更新数据库
119        db.close();
120    }
121
122    //方法:添加日志
123    public void insertDiary(){
124        SQLiteDatabase db = myHelper.getWritableDatabase();                   //获得数据库对象
125        ContentValues values = new ContentValues();
126        values.put(TITLE, etModifyTitle.getText().toString());
127        values.put(CONTENT, etModifyContent.getText().toString(ry);
128        values.put(FACE, faceselectedid);
129        values.put(DATETIME, tvdttm.getText().toString());
130        db.insert(TABLE_NAME, ID, values);                                    //插入数据
131        db.close();
132    }
133 }
```

① 增加了第 33 行一条赋值语句，在心情图标 Spinner 控件的 Adapter 监听器中，获取选中的索引号。

② 第 40 行创建一个 SQLiteOpenHelper 对象，打开数据库表。

③ 第 42 行获得 Intent 对象的附加信息 cmd 的值并赋给变量 status，确定用户发出的是编辑还是新增数据表记录。第 45～64 行定义查看或修改日志的代码，当 status 值为 0 时，第 46 行获得用户选中的记录 ID 号，第 48～51 行执行对数据表的每一列，日志 ID 号为用户所选 ID 记录的查询；如果找到需要的日志，在第 56～60 行从数据表中取出相应信息显示在屏幕的对应控件中；第 62、63 行关闭游标和数据库。当 status 值为 1 时，清空屏幕上的用于显示日志标题、日志日期和日志内容等控件，心情图标默认为第一个，为新增一条日志作准备。

④ 第 73～98 行定义"发布日志"按钮的监听。在其中，第 79 行检查日志的标题编辑框和内容编辑框内容是否有为空的，如果有一个为空，用户必须回去填写；第 84 行按指定的格式获得系统当前的日期时间值；第 85～92 行根据 status 值，分别调用自定义方法 updateDiary()和 insertDiary()。

⑤ 第 111～120 行定义了 updateDiary()方法，其中第 118 行调用 SQLiteDatebase 的 update()方法修改指定 ID 的记录内容。

⑥ 第 123～132 行定义了 insertDiary()方法，其中第 130 行调用 SQLiteDatebase 的 insert()方法添加一条新记录内容。

⑦ 在定义"返回"按钮的监听中，增加了第 104 行，将当前日志的 ID 号附加到返回的 Intent 中。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 KDWB_Diary2 项目。按下手机(或模拟器)中的 menu 键，调出选项菜单，如图 7-4 所示。当单击"添加"菜单项，进入日志编辑窗口进行添加新日志操作，完成后可单击"发布日志"按钮，将录入的新日志信息保存到数据库表中并返回到日志列表窗口，如图 7-5 所示。



图 7-4　调出选项菜单界面



图 7-5　日志录入编辑界面

到此为止，本章所介绍的应用程序数据存储方式，无论是使用配置文件、文件存储还是数据库存储，数据是应用程序私有的。在 Android 系统中，没有一个公共的内存区域，供多个应用共享存储数据，所以应用程序之间是相互独立的，它们分别运行在自己的进程中。而 ContentProvider 机制可以支持在多个应用程序中存储和读取数据，这个组件是解决跨应用共享数据的唯一方式。

7.4 ContentProvider

ContentProvider 是所有应用程序之间数据存储和检索的桥梁，它的作用就是使得各个应用程序之间实现数据共享。在 Android 中，ContentProvider 是一种特殊的存储数据的类型，它提供了一套标准的接口用来获取、操作数据。例如音频、视频、图片和个人联系信息等几种常用的 ContentProvider。它们被定义在 android. provider 包下。通过这些定义好的 ContentProvider 可以方便地进行数据读取，也可以进行数据删除等操作。当然，执行这些操作必须拥有适当的权限，记住，要在应用项目的 AndroidManifest. xml 中把相关权限添加进去。

7.4.1 实现数据共享的相关类、接口与权限

1. ContentProvider

ContentProvider 类位于 android. content 包下。一个程序可以通过实现一个 ContentProvider 的抽象方法接口将自己的数据暴露出去，外部程序可以通过一组标准的接口来和本程序内的数据打交道。所以定义一个 ContentProvider 必须要实现几个抽象方法接口，这些常用抽象方法如表 7-7 所示。

表 7-7 ContentProvider 的常用抽象方法及说明

方 法	描 述
query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)	通过 URI 进行查询,返回一个 Cursor
insert(Uri uri,ContentValues values)	将一组数据插入到 URI 指定的地方
delete(Uri uri,String where, String[] selectionArgs)	删除指定 URI 并且符合一定条件的数据
update(Uri uri, ContentValues values, String where, String[] selectionArgs)	更新 URI 指定位置并且符合一定条件的数据
getType(Uri uri)	获得 MIME 数据类型
onCreate()	当 ContentProvider 创建时调用
getContext()	获得 Context 对象

在上述方法中使用最多的是 query()方法,下面对其参数进行说明。

✍参数 uri 为指定的 URI 地址。

✍参数 projection 为指定的返回列名。

✍参数 selection 用于指定返回的行,相当于 SQL 语句中的 WHERE 子句。

✍参数 selectionArgs 对参数 selection 中出现的"?"进行替换。

✍参数 sortOrder 指定返回结果的排序方式。

2. ContentResolve

ContentResolve 类位于 android.content 包下。当外部应用程序需要对 ContentProvider 中的数据进行添加、删除、修改和查询操作时,可以使用 ContentResolver 接口来完成,在 Activity 中通过 getContentResolver()方法获取 ContentResolver 对象。ContentResolver 提供的抽象方法与 ContentProvider 需要实现的方法对应,同样使用 query()、insert()、delete()、update()等方法来操作数据。这些方法的名称和参数与 ContentProvider 中的一样,在此不作赘述。

一般情况下,ContentProvider 是单实例的,但是可以有多个 ContentResolve 在不同的应用程序和不同的进程之间和 ContentProvider 交互。

3. URI

URI 是指向数据的一个资源标识符。通过 URI 使得 ContentResolve 知道与哪一个 ContentProvider 对应,并且来操作哪些数据。在 ContentProvider 和 ContentResolve 中 URI 通常有两种形式,一种是指定全部数据,另一种是指定某个 ID 数据。例如:

1) 指定全部的联系人数据

```
content://contacts/people/
```

2) 指定 ID 为 2 的联系人数据

```
content://contacts/people/2
```

通常 URI 比较长,在编程时容易出错,且难以理解。所以在 Android 中定义了一些辅助类,在其中定义了一些常量来代替这些长字符串的 URI。

例如,在定义 ContentProvider 接口的代码文件 DiaryContentProvider.java 中定义:

```
public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/diaries");
```

此后 CONTENT_URI 相当于“content://cn.com.sgmsc.ZSWB.DiaryContentProvider/diaries”。

4. 设置权限

需要注意的是，无论对哪一个 ContentProvider 中的数据进行操作，进行哪一类操作，都需要在 AndroidManifest.xml 文件中添加相应的权限。例如要对手机的通讯录进行查询和修改操作，则在 AndroidManifest.xml 文件的＜manifest＞标签内需要添加下列权限设置：

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
```

7.4.2 ContentProvider 应用案例

下面通过一个简单案例来说明 ContentProvider 如何实现应用程序之间的数据共享。

【案例 7.3】 自定义一个 Activity，用于查看手机通讯录中的信息。

【说明】 每一个手机都定制有通讯录管理应用程序。在 SDK 2.3 以前的版本中，这个应用程序定义于 android.provider.Contacts.Phones 类中，系统定义 Phones.CONTENT_URI 常量来表示通讯录的数据 URI。在 SDK 2.3 以后的版本中，该应用程序定义于 android.provider.ContactsContract 类中，系统定义 ContactsContract.CommonDataKinds.Phone.CONTENT_URI 常量来表示通讯录数据的 URI。

本例将使用 SimpleCursorAdapter 为 ListView 绑定数据。SimpleCursorAdapter 中的数据取自于一个游标对象。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 ContentProvider 的 Android 项目。其应用程序名为 ContentPrd，包名为 cn.com.sgmsc.CP，Activity 组件名为 ContentPrdActivity。

(2) 开发逻辑代码。本例不修改项目创建时生成的布局文件 main.xml，只对 Java 代码文件作修改。编辑 src/cn.com.sgmsc.CP 包下的 ContentPrdActivity.java，代码如下所示。

```
package cn.com.sgmsc.CP;

import android.app.ListActivity;
import android.database.Cursor;
import android.os.Bundle;
import android.content.ContentResolver;
import android.provider.ContactsContract;
import android.widget.ListAdapter;
import android.widget.SimpleCursorAdapter;

public class ContentPrdActivity extends ListActivity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ContentResolver cr = getContentResolver();
```

```
17            Cursor c = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null, null,
              null, null);
18           startManagingCursor(c);
19           ListAdapter adapter = new SimpleCursorAdapter(this,
20                   android.R.layout.simple_list_item_2, c,
21                           new String[] { ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
22                                       ContactsContract.CommonDataKinds.Phone.NUMBER },
23                           new int[] { android.R.id.text1, android.R.id.text2 });
24           setListAdapter(adapter);
25       }
26
27 }
```

① ContentPrdActivity 是继承 ListActivity 类，所以在第 3 行引入 android. app. ListActivity 类。

② 第 16 行，使用 getContentResolver()方法得到一个 ContentResolver 实例 cr。

③ 第 17 行使用 cr. query()查询手机通讯录中所有联系人信息，并返回到一个 Cursor 对象 c 中。其中 ContactsContract. CommonDataKinds. Phone. CONTENT_URI 是指向全体联系人的 URI。

④ 第 18 行，startManagingCursor(c)语句是让系统来管理生成的 Cursor。

⑤ 第 19～23 行，ListAdapter adapter = new SimpleCursorAdapter(…)是从 Cursor 对象 c 中取出数据生成一个 ListAdapter 适配器对象 adapter。其中第一个参数 this 表示当前上下文的引用；第二个参数 android. R. layout. simple_list_item_2，表示这个 ListAdapter 每一个条目的显示布局由系统定义的布局文件确定；第三个参数 c 表示数据集来源于 Cursor 对象 c；第四个参数是一个 String 数组：new String[] {…. DISPLAY_NAME，…. NUMBER }，其中 ContactsContract. CommonDataKinds. Phone. DISPLAY_NAME 用于从通讯录中的联系人姓名列中取值，ContactsContract. CommonDataKinds. Phone. NUMBER 用于从通讯录中的联系人电话号码列中取值；第五个参数是一个 int 数组：new int[] { android. R. id. text1，android. R. id. text2 }，该数组的值是 android. R. layout. simple_list_item_2 中定义的两个 TextView 的 ID。

⑥ 第 24 行，setLiAdapter(adapter)将 ListView 和 SimpleCursorAdapter 进行绑定，并显示出来。

(3) 编辑 AndroidManifest. xml 文件。修改项目根目录下的 AndroidManifest. xml 文件，添加相应权限，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "cn.com.sgmsc.CP"
    android:versionCode = "1"
    android:versionName = "1.0" >

    <uses - sdk android:minSdkVersion = "10" />
    <uses - permission android:name = "android.permission.READ_CONTACTS" />

    <application
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name" >
```

```
13         <activity
14             android:label = "@string/app_name"
15             android:name = ".ContentPrdActivity" >
16             <intent - filter >
17                 <action android:name = "android.intent.action.MAIN" />
18
19                 <category android:name = "android.intent.category.LAUNCHER" />
20             </intent - filter >
21         </activity>
22     </application>
23
24 </manifest>
```

第 8 行，添加对手机通讯录数据的可读权限。

【运行结果】 如果你的手机或模拟器的通讯录中没有任何联系人信息，请先向通讯录中添加若干个联系人。其操作步骤如下。

(1) 按手机(或模拟器)中的 Home 键，找到桌面中的 Contacts 应用程序，如图 7-6 所示。

(2) 单击 Contacts 进入应用程序，然后按下手机(或模拟器)中的 menu 键，在选项菜单中选择 New contact 菜单项，对联系人姓名、电话号码等信息进行添加。

(3) 重复第(2)步多添加几个联系人，如图 7-7 所示。然后退出 Contacts 应用。

现在可以运行本案例了，运行 ContentProvider 项目，即可在屏幕上看到通讯录中的信息，如图 7-8 所示。



图 7-6 模拟器桌面中的应用程序

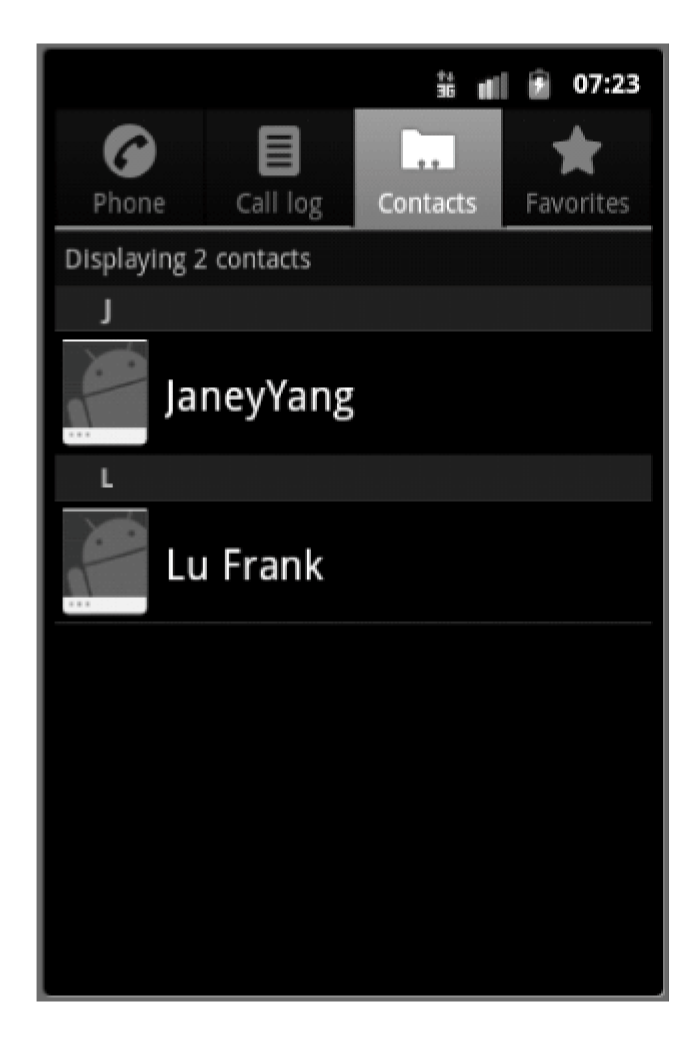


图 7-7 Contacts 应用程序的联系人列表

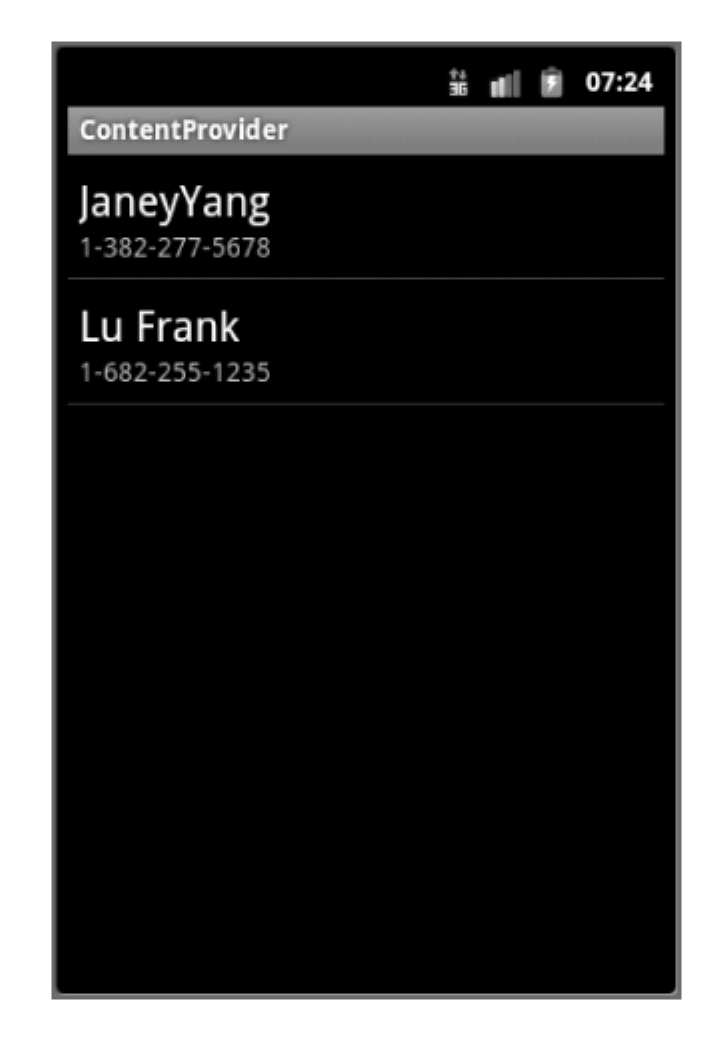


图 7-8 在 ContentProvider 中看到联系人信息

7.5 访问 SD 卡简介

手机除了本机内存外，还可以使用 SD 卡来扩展存储空间。SD 卡(Secure Digital Memory Card，安全数码卡)是一种基于半导体快闪记忆器的新一代记忆便携式设备，拥有高记忆容量、快速数据传输率、极大的移动灵活性以及很好的安全性。目前 Android 支持 8MB～128GB 的

SD 卡。

在使用模拟器开发时，可以通过硬盘来模拟 SD 卡。模拟 SD 卡的步骤是：第一步创建一个 SD 卡镜像文件；第二步关联 SD 卡和模拟器；第三步向 SD 卡中导入文件；最后是在模拟器中使用 SD 卡中的文件。下面对每一步骤作详细介绍。

1. 创建一个 SD 卡镜像文件

创建 SD 卡是在命令提示符状态下完成的。创建 SD 卡的命令文件名为 mksdcard. exe，存放在 Android SDK 的安装文件夹的 tools 子文件夹下，例如在本书中，创建 SD 卡命令 mksdcard. exe 位于 E:\android-sdk \tools 文件夹下。为了避免运行此文件时需要输入一长串的路径，建议先设置 Windows 操作系统的环境参数 path，在 path 中添加设置 sdk 文件夹下的 tools 文件夹路径(如果是 Android 3. 0 还要设置 platform-tools 文件夹路径)。注意，本书后续章节都是在设置了该路径的环境下进行阐述，使用时不再提及。在做好上述系统环境参数设置后，就可以按如下步骤创建 SD 卡镜像文件了。

(1) 运行 cmd 命令，进入 DOS 命令行提示状态。

(2) 确定存放 SD 卡镜像文件存放位置(用 DOS 命令进入存放镜像文件的子目录中)。例如，在 DOS 命令符状态下输入 DOS 命令"CD C:\Users\compaq\. android\avd"。注意，mksdcard 命令不会自动建立不存在的文件夹，因此，在执行上面命令之前要确认指定文件夹已存在，否则先创建之。

(3) 输入如下 DOS 命令"mksdcard 1024M sdcard. img"。其中，mksdcard 是创建命令；1024M 是 SD 卡容量，即为 1GB，这里的单位为 K、M(必须大写)；sdcard. img 是 SD 卡镜像文件名，在当前文件夹中(可以指定路径)。

(4) 完成上述操作后就可以在指定文件夹下看到刚建立的 sdcard. img 文件了。

2. 关联 SD 卡和模拟器

关联 SD 卡和模拟器实际上是让模拟器知道它的 SD 卡镜像文件存储的路径。这个关联操作可使用三种方式，第一种是在 DOS 命令行提示状态下使用命令方式，另外两种是在 Eclipse 中使用菜单进行可视化设置。下面分别介绍。

1) 方式一

运行 cmd 命令，进入 DOS 命令行提示状态，输入命令：

```
emulator - avd AVD_and - 23 - sdcard c:\users\compaq\.android\avd\sdcard.img
```

该命令可启动 Eclipse 的 AVD，并自动为其绑定指定的 SD 卡。其中"AVD_and-23"是模拟器的名称，"c:\users\compaq\. android\avd\sdcard. img"是 SD 镜像文件的全路径文件名。

2) 方式二

(1) 打开 Eclipse，选择菜单 Run→Run Configurations...，进入 Run Configurations 窗口。

(2) 在右边窗口的选项卡中选择 Target，如果有多个模拟器，请在 Select a preferred Androd Virtual Device for deployment 复选框中选择需要的模拟器。

(3) 在 Target 选项页下方的 Additional Emulator Command Line Options 的编辑框中输入"-sdcard c:\users\compaq\. android\avd\sdcard. img"，这就是模拟器启动时的参数，如图 7-9 所示。

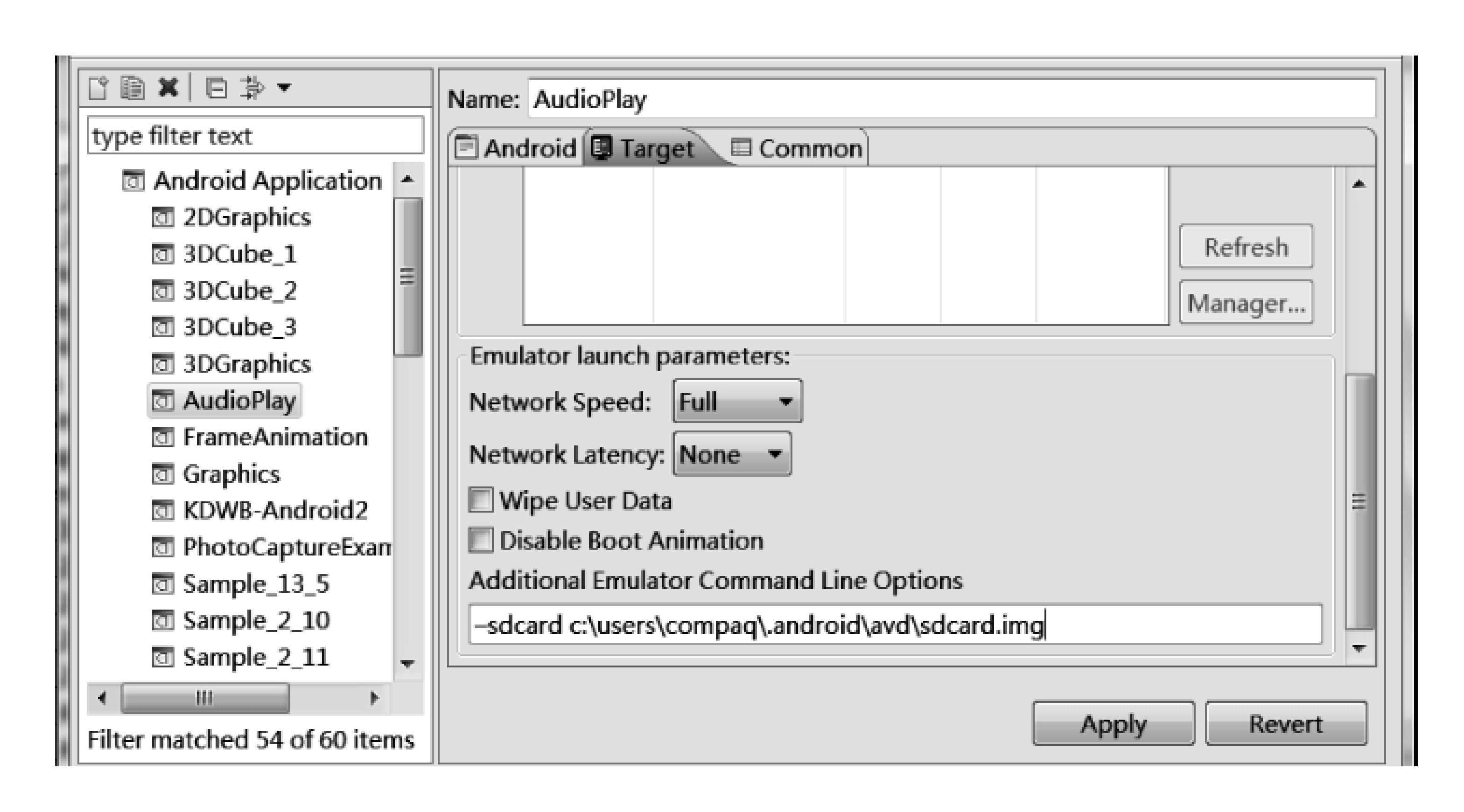


图 7-9 设置模拟器启动时的参数

(4) 单击 Apply 按钮即完成设置。

3) 方式三

(1) 打开 Eclipse,选择菜单 Window→Preferences,进入 Preferences 窗口。

(2) 在 type filter text 中选择 Android→Launch,在右边的 Default emulator options 的编辑框中输入"-sdcard c:\users\compaq\.android\avd\sdcard.img",设置模拟器默认的启动参数,如图 7-10 所示。

(3) 单击 Apply 按钮即完成设置。

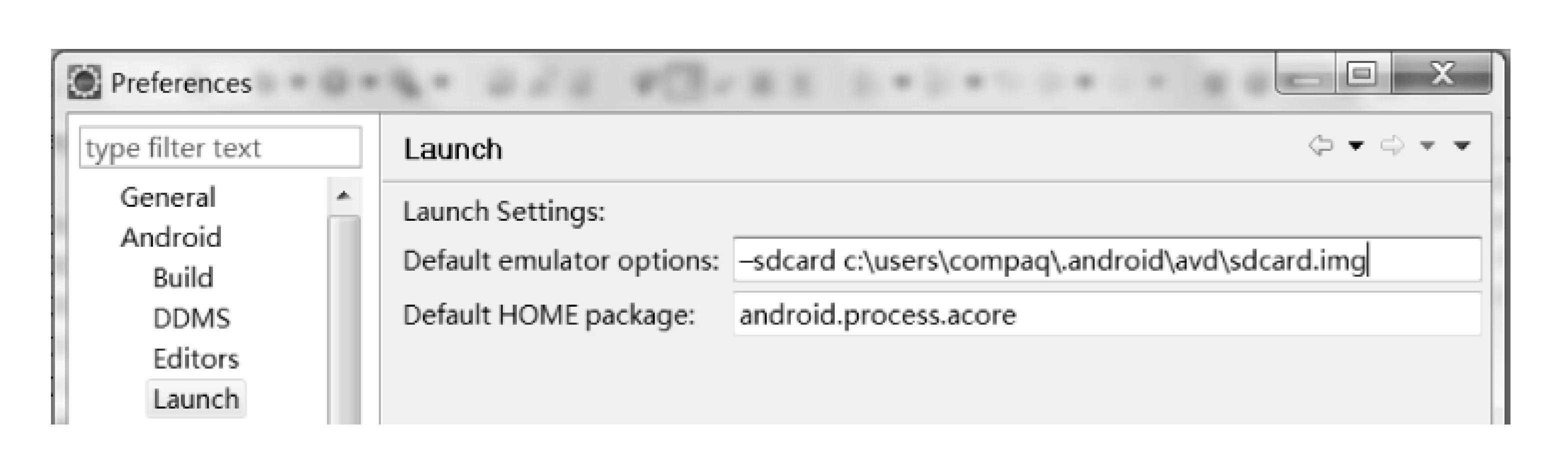


图 7-10 设置模拟器的默认启动参数

3. 向 SD 卡导入文件

初始创建的 SD 卡内没有任何文件,需要向其内装载文件。可以按如下步骤向 SD 卡镜像文件中装载文件。

(1) 在 Eclipse 中启动模拟器。

(2) 向 SD 卡镜像文件中装载文件。有两种方式向 SD 卡导入文件,一种是在 DOS 命令行提示状态下使用命令方式,另一种是在 Eclipse 中使用菜单进行可视化设置。下面分别介绍。

① 方式一。

运行 cmd 命令,进入 DOS 命令行提示状态,输入命令:

```
adb push e:\3gms\music02.mp3  /sdcard/music.mp3
```

该命令将指定目录下的 music02.mp3 音频文件导入到名为 sdcard.img 的镜像文件中,即

导入到了 SD 卡中。注意,sdcard.img 可以不在音频文件的文件夹中。这里,music02.mp3 文件是在 e:\3gms 文件夹中,而 sdcard.img 则在 c:\users\compaq\.android\avd 文件夹中,导入到 SD 卡中的文件名为 music.mp3。

② 方式二。

✍在 Eclipse 中,打开 DDMS 的 FileExplorer 选项卡。

✍选择 sdcard,单击右上角的 push 按钮,打开 Put File on Device 窗口,选择要导入的文件,然后单击"打开"按钮,将文件导入进来。

✍导入后即可在 sdcard 目录下看到刚导入的文件,如图 7-11 所示。

Threads | Allocation Tracker | Heap | File Explorer

Name	Size	Date	Time	Permissions	Info
data		2011-04-14	10:10	drwxrwx--x	
mnt		2011-05-05	14:25	drwxrwxr-x	
asec		2011-05-05	14:25	drwxr-xr-x	
obb		2011-05-05	14:25	drwxr-xr-x	
sdcard		2011-05-05	14:29	d---rwxr-x	
LOST.DIR		2011-04-12	02:33	d---rwxr-x	
music.mp3	1278352	2011-05-04	09:18	----rwxr-x	
secure		2011-05-05	14:25	drwx------	
system		2011-01-20	00:51	drwxr-xr-x	

图 7-11 在 DDMS 的 File Explorer 中查看 SD 卡中的文件

4. 使用 SD 卡中的文件

在向 SD 卡中导入了文件之后,在应用程序中就可以使用这些文件了。例如,向 SD 卡中导入了一个音频文件 music. mp3,可以在代码文件中编写如下代码。

```
import android.media.MediaPlayer;                //引入媒体播放器类
……
MediaPlayer mp = new MediaPlayer();              //创建一个 MediaPlayer 对象 mp

try{
    mp.setDataSource("/sdcard/music.mp3");       //设置音乐文件取自于 SD 卡中的 music.mp3
    mp.prepare();
    mp.start();                                  //播放音乐
}
catch(Exception e){
     e.printStackTrace();
}
```

小结

本章重点介绍了应用程序的数据的 SharedPreferences 存取和 SQLite 数据库操作的方法技术,还介绍了应用程序之间的数据共享技术 ContentProvider 和使用 SD Card 存取文件的方法步骤。掌握了这些数据存储技术和数据共享技术,加之 Java 的文件读写处理技术,要开发一个 Android 的本地应用就会毫无问题。

在介绍 SQLite 数据库操作时涉及数据库的数据描述语言 DDL 和数据操纵语言 DML 相关内容,这些也是开发应用程序时必备的数据库技术。

为了让用户界面更炫更精彩,为了让用户能更好地体验应用程序的操作使用,可以向应用

程序中加入更多的表现形式，如在应用中添加一段音乐、插入一个视频等。第 8 章将介绍 Android 的多媒体应用开发。



练习

设计“掌上微博”的联系人查看及好友保存模块，要求至少使用两个 Activity，一个 Activity 用于列表显示好友名单，另一个 Activity 是显示选定好友的详细信息，并保存到本地数据库中。

第8章 多媒体应用开发

多媒体的应用是现今应用项目中不可缺少的增值亮点。在前面的章节中，已经学习了对于文本、图片的处理运用，本章将进一步学习对图形、动画、音频和视频等多媒体的应用开发。

8.1 2D、3D 图形

Android 系统的图形处理能力非常强大，它自定义了一系列 2D 图形处理类，这些类分别位于 android. graphics、android. graphics. drawable. Opshapes 和 android. view. animation 包中。对于 3D 图形的处理，Android 集成了 OpenGL ES 1.0 提供的高效 3D 图形处理技术，这些类分别位于 javax. microedition. khronos. opengles 和 android. opengl 包中。下面分别介绍。

8.1.1 2D 图形相关类

Android 在其 android. graphics 包中提供了完整的本机的 2D 图形库。在 android. graphics 包中常用类有：Color 类、Paint 类、Canvas 类、Path 类和 Drawable 类。

1. Color 类

Android 中的颜色用 4 个数字表示，它们分别指定透明度、红色、绿色、蓝色(Alpha、Red、Green、Blue，ARGB)所占的比重，每个数字取值在 0～255 之间(如果使用二进制数制表示，则占 8 位)。由于每个数字占 8 位，因此一种颜色通常需要 32 位整数来表示。为了编程方便，常常使用十六进制的数字来表示颜色。

红、绿、蓝的数字大小代表颜色的深浅度，数字越小颜色越深，数字越大颜色越浅。但是透明度则不然，透明度取 0 时表示完全透明，取 255 时表示颜色完全不透明。

在开发中，如果可能，最好在一个 XML 资源文件中定义应用所需的颜色。这样，以后可以在一个地方轻松地更改这些颜色定义，就可以使得运用到这种颜色的地方发生改变而不需要到处对代码进行修改了。在前面的章节中已经多次使用 XML 资源文件定义颜色。例如在 XML 文件中定义一个紫红色的名为 mycolor 的颜色，代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
```

```
        <color name = "mycolor">#7fff00ff</color>
</resources>
```

这时，可在 Java 代码中通过名称来引用颜色，例如代码：

```
Int color = getResources().getColor(R.color.mycolor);
```

代码中，getResources()方法返回当前 Activity 的 ResourceManager 类，getColor()方法是要求管理器根据资源 ID 查找指定的颜色。

如果没有在 XML 文件中定义颜色，可以在 Java 代码文件中使用十进制数字来定义颜色，或使用系统预定义的标准颜色。代码如下。

```
color = Color.argb(127,255,0,255);          //使用十进制数表示紫红色
color = Color.BLUE;                         //使用系统预定义的蓝色颜色值
```

2. Paint 类

Paint 类是 Android 本机图形库中最重要的类之一。它包含样式、颜色以及绘制任何图形所需的其他信息。Paint 类常用方法如表 8-1 所示。

表 8-1　Paint 类常用的方法及说明

方　　法	返回值	描　　述
setARGB(int a, int r, int g, int b)	void	设置 Paint 对象颜色
setAlpha(int a)	void	设置 alpha 不透明度，范围为 0～255
setColor(int color)	void	设置颜色，这里 Android 内部定义的有 Color 类，包含一些常用颜色定义
setTextAlign(Paint. Align align)	void	设置文本对齐
setTextSize(float textSize)	void	设置字体大小
setTypeface(Typeface typeface)	Typeface	设置字体，Typeface 包含字体的类型、粗细，还有倾斜、颜色等

例如在 Java 代码中设置浅灰色的预定义颜色，可使用 setColor()方法，代码如下。

```
myPaint.setColor(Color.LTGRAY);
```

3. Canvas 类

Canvas 类代表可以在其上绘图的画布。最初，画布启动时上面没有任何内容，就像一张白纸一样。利用 Canvas 类中的各种方法可以在画布上绘制线条、矩形、圆以及其他任意图形。

Android 中的显示屏幕是由 Activity 类的对象表现的，而 Activity 类的对象引用的是 View 类的对象，而 View 类的对象又是引用 Canvas 类的对象。这一系列的类引用，通过重写 View. onDraw()方法，可以在指定的画布上绘图。onDraw()方法只有一个参数，它用于指定所在的画布对象。Canvas 类常用的方法如表 8-2 所示。

表 8-2 Canvas 类常用的方法及说明

方法	描述
drawText(String text, float x, float y, Paint paint)	在屏幕上描绘文字,参数 text 是 String 类型的文本,参数 x 为水平轴坐标,参数 y 为垂直轴坐标,参数 paint 为画刷对象
drawTextOnPath(String text, Path path, float x, float y, Pating paint)	在屏幕上沿着图形的轨迹描绘文字
drawPoint(float x, float y, Pating paint)	画点,参数 x 为水平轴坐标,参数 y 为垂直轴坐标,参数 paint 为画刷对象
drawLine(float startX, float startY, float endX, float endY, Pating paint)	画线,参数 startX 为起始点的 x 坐标,参数 startY 为起始点的 y 坐标,参数 endX 为终点的 x 坐标,参数 eyndY 为终点的 y 坐标,参数 paint 为画刷对象
drawCircle(float cx, float cy, float radius, Pating paint)	画圆,参数 cx 是圆心点的 x 坐标,参数 cy 是圆心点的 y 坐标,参数 radius 是半径,参数 paint 为画刷对象
drawOval(RectF oval, Pating paint)	画椭圆,参数 oval 为一个区域
drawRect(RectF oval, Pating paint)	画矩形,参数 oval 为一个区域
drawPath(Path path, Paint paint)	画一个路径,参数 path 为路径对象

4. Path 类

Path 类包含一组矢量绘图命令,例如画线条、画矩形和画曲线等。例如画一个圆形,其圆心坐标为 x=160,y=180;半径为 118 像素;绘制方向为顺时针。实现的代码如下。

```
circle = new Path();
circle.addCircle(160,180,118,Direction.CW);
```

这里,Direction.CW 为顺时针方向,而 Direction.CCW 是逆时针方向。

5. Drawable 类

Android 中的 Drawable 类主要针对位图或纯色的可视元素,如按钮、视图背景等。Drawable 类支持的格式如表 8-3 所示。

表 8-3 Drawable 类支持的格式及描述

格式	描述
Bitmap(位图)	PNG 或 JPEG 图像
NinePatch(九宫格)	一种可扩展的 PNG 图像,主要用作大小可调整的位图按钮的背景
Shape(形状)	基于 Path 类的矢量绘图命令
Layers(图层)	盛放子绘图区的容器
States(状态)	一个容器,主要用于选择不同的按钮并设置按钮的焦点状态
Levels(级别)	一个容器,主要用于电池或信号强度指示器
Scale(缩放)	包含一个子绘图区的容器,能根据可绘图区的当前级别调整其大小。用于可缩放的图片查看器

可绘图区一般都在 XML 文件中定义。例如,定义一个颜色从浅绿到果绿渐变的可绘图区,且渐变的方向是从上向下的,则在 XML 中代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
    < gradient
        android:startColor = "#ffe9f5db"
        android:endColor = "#ffabdb77"
        android:angle = "270"/>
</shape>
```

这里，标签< gradient >定义渐变色，属性 startColor 为开始颜色，endColor 为结束颜色，angle 为渐变方向，其值所代表的方向如表 8-4 所示。

表 8-4　属性 angle 的取值及对应的渐变方向

值	渐变显示方式	值	渐变显示方式
0	开始色在左，结束色在右的横向渐变	90	开始色在下，结束色在上的纵向渐变
180	开始色在右，结束色在左的横向渐变	270	开始色在上，结束色在下的纵向渐变

使用这个可绘图区的方法有两种：一是在 XML 中以 android:background＝属性这种形式引用它；二是在视图的 onCreate()方法中调用 Canvas.setBackgroundResource()方法。

8.1.2　绘制 2D 图形案例

【案例 8.1】　在渐变的背景上画一空心圆，圆内以环形的方式显示文字串。

【说明】　定义背景色的 XML 文件是存放在 res/drawable 目录下的。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 2DGraphics 的 Android 项目。其应用程序名为 GraphicsDemo，包名为 cn.com.sgmsc.Graphics，Activity 组件名为 GraphicsActivity。

(2) 创建颜色资源。在 res/values 目录下创建一个 colors.xml 文件，用于定义渐变背景色的开始和结束颜色，定义文字的颜色。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <color name = "startcl">#ffe9f5db</color>
    <color name = "endcl">#ffabdb77</color>
    <color name = "mycolor">#7fff00ff</color>
</resources>
```

(3) 创建背景资源。在图片资源目录 res/drawable-mdpi 下创建一个 background.xml 文件，用于定义屏幕的背景色。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
    < gradient
        android:startColor = "@color/startcl"
        android:endColor = "@color/endcl"
        android:angle = "270" />
</shape>
```

(4) 设计布局。res/layout 目录下的 main.xml 文件很简单，因为背景色和绘图、写文字都在其他的文件中设置，所以在这个布局文件中只需要定义一个线性布局层就可以了。代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
</LinearLayout>
```

(5) 开发逻辑代码。打开 src/cn.com.sgmsc.Graphics 包下的 GraphicsActivity.java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.Graphics;

import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Path.Direction;
import android.os.Bundle;
import android.view.View;

public class GraphicsActivity extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(new GraphicsView(this));
   }

   static public class GraphicsView extends View {
      private static final String CIRCLETXT = "Android应用开发教程 - 第一版.2012.12";
      private final Path circle;
      private final Paint cir_Paint;
      private final Paint txt_Paint;

      public GraphicsView(Context context) {
          super(context);
          circle = new Path();
          circle.addCircle(160, 180, 118, Direction.CW);
          //定义画圆的相关属性: 消锯齿、空心圆、颜色、线粗细
          cir_Paint = new Paint(Paint.ANTI_ALIAS_FLAG);
          cir_Paint.setStyle(Paint.Style.STROKE);
          cir_Paint.setColor(Color.argb(255, 202, 232, 170));
          cir_Paint.setStrokeWidth(6);
          //定义圆内环形文本的相关属性: 消锯齿、实心字、颜色、字号
          txt_Paint = new Paint(Paint.ANTI_ALIAS_FLAG);
          txt_Paint.setStyle(Paint.Style.FILL_AND_STROKE);
          txt_Paint.setColor(Color.GRAY);
          txt_Paint.setTextSize(20f);
          setBackgroundResource(R.drawable.background);//定义渐变背景色
      }
      @Override
      protected void onDraw(Canvas canvas) {
          canvas.drawPath(circle, cir_Paint);
          canvas.drawTextOnPath(CIRCLETXT, circle, 0, 26, txt_Paint);
```

```
46          }
47      }
48 }
```

① 第 20～47 行定义一个静态的内部类 GraphicsView，它继承自 View 类。这个类包含一个 Canvas。在这个类中定义了两个方法：GraphicsView()和 onDraw()，onDraw()在 GraphicsView()初始化完成之后开始调用。

② 第 26～41 行定义的是 GraphicsView()方法。其中，第 28、29 行创建一个矢量图形对象 circle，这个 circle 是一个圆形对象，其圆心坐标为 x=160，y=180；半径为 118 像素；绘制方向为顺时针。第 31～34 行设置图形对象 cir_Paint 的若干属性值：消锯齿、空心圆、颜色、线粗细。第 36～39 行设置图形对象 txt_Paint 的若干属性值：消锯齿、实心字、颜色、字号。第 40 行设置画面绘图区的背景色。

③ 第 43～46 行定义了 onDraw()方法。第 44 行按照 cir_Paint 指定的属性画出 circle 来；第 45 行沿着 circle 的轨迹描绘文字，文字内容取自 CIRCLETXT 常量，且文本的属性由 txt_Paint 指定。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 2DGraphics 项目，显示效果如图 8-1 所示。



图 8-1　本例运行效果

8.1.3　3D 图形编程

本节将对 Android 平台下的 3D 编程进行简单介绍，并通过一些案例来说明如何使用 GLSurfaceView 及 OpenGL ES 标准来开发 3D 应用程序。由于 3D 编程涉及的知识非常多，这里无法一一涵盖，所以只是对其原理和开发流程作简单的介绍。

1. OpenGL ES 简介

Android 通过 OpenGL ES API，可以支持高性能的 3D 图形，而 OpenGL ES 是 OpenGL 的子集。

OpenGL 是个专业的 3D 图形软件接口标准，是由 SGI 公司开发的一套功能强大、调用方便的底层 3D 图形库。OpenGL ES(为 OpenGL for Embedded System 的缩写)是为适用于嵌入式设备，根据 OpenGL 规范进行裁剪后，形成的一套精简标准。其官方网址为 http://www.khronos.org。它适用于手机、PDA 或其他移动终端的 3D 显示应用。

OpenGL ES 1.0 和 1.1 API 规范从 Android 1.0 就开始支持了。自 Android 2.2(API Level 8)以后，框架支持 OpenGL ES 2.0 API 规范，OpenGL ES 2.0 被大多数 Android 设备所支持并被推荐在新的基于 OpenGL 的应用中使用。Android 2.0 版本之后图形系统的底层渲染均由 OpenGL 负责，OpenGL 除了负责处理 3D API 调用，还需负责管理显示内存及处理 Android SurfaceFlinger 或上层应用对其发出的 2D API 调用请求。注意，OpenGL 2.0 向下兼容到 OpenGL 1.5，而 OpenGL ES 2.0 和 OpenGL ES 1.x 不兼容，是两种完全不同的实现。

Android 提供基于 OpenGL ES 标准的三维图形库，其 3D 图形系统分为 Java 框架和本地

代码两部分。本地代码主要实现OpenGL接口的库，在Java框架层，其标准的OpenGL包是javax.microedition.khronos.opengles，而android.opengl包则提供OpenGL系统和Android GUI系统之间的联系。Android支持OpenGL列表有GL、GL 10、GL 10 EXT、GL 11等，这些GL都是公共接口，它包含Java™程序语言为OpenGL绑定的核心功能。本书将使用GL10这个类来接触OpenGL，探索3D领域。

2. 3D开发基本

1) 3D图形的构成

构成3D图形的最基本的单位是顶点(Vertex)，它代表空间中的一个点。通过若干个点可以构成多边形，通过多边形又可以构成更复杂的空间物体。OpenGL可以支持多种多边形，而OpenGL ES所支持的基本图形只有点(Point)、线(Line)和三角形(Triangle)，任何其他的3D图形都是通过这几种基本几何图形组合而成。

在OpenGL ES中，表示顶点由三个数值确定，分别表示x、y、z三个坐标值。直线由两个顶点索引值表示。三角形由三个顶点索引值表示。表示一个点的颜色时，由4个数值确定，分别表示色彩值R、G、B、A的值。

一般将构成空间物体的顶点的坐标值以顶点数组的形式给出。顶点数组包括3D场景中部分或全部的顶点坐标数据，其中每三个数组元素表示一个顶点的坐标。例如，场景中有三个顶点A(1,2,3)、B(4,5,6)、C(7,8,9)，则其顶点数组为{1,2,3,4,5,6,7,8,9}。类似地，三角形、顶点颜色也是由数组来描述的。

2) 绘制3D图形的方法

OpenGL ES提供了两类方法来绘制一个空间几何图形。一个是glDrawArrays()，另一个是glDrawElements()。

public abstract void glDrawArrays(int mode, int first, int count) 使用VetexBuffer来绘制，顶点的顺序由vertexBuffer中的顺序指定。

public abstract void glDrawElements(int mode, int count, int type, Buffer indices)可以重新定义顶点的顺序，顶点的顺序由indices指定。这里，mode值可取如下常量：GL_POINTS(绘制独立的点)、GL_LINE_STRIP(绘制一条线段)、GL_LINE_LOOP(绘制一条封闭线段，即首尾相连)、GL_LINES(绘制多条线段)、GL_TRIANGLES(绘制多个三角形，且两两不相邻)、GL_TRIANGLE_STRIP(绘制多个三角形，且两两相邻)、GL_TRIANGLE_FAN(以一个点为顶点绘制多个相邻的三角形)等。

对应顶点，除了可以为其定义坐标外，还可以指定颜色、材质、法线(用于光照处理)等。glEnableClientState和glDisableClientState可以控制的管道(Pipeline)开关有：GL_COLOR_ARRAY (颜色)、GL_NORMAL_ARRAY(法线)、GL_TEXTURE_COORD_ARRAY(材质)、GL_VERTEX_ARRAY(顶点)等。相关的方法如下。

(1) glColorPointer(int size,int type,int stride,Buffer pointer)，传入颜色。

(2) glVertexPointer(int size, int type, int stride, Buffer pointer)，传入顶点。

(3) glTexCoordPointer(int size, int type, int stride, Buffer pointer)，传入材质。

(4) glNormalPointer(int type, int stride, Buffer pointer)，传入法线。

OpenGL ES内部存放图形数据的Buffer有COLOR、DEPTH (深度信息)等，在绘制图形之前一般需要清空COLOR和DEPTH的Buffer。

3）3D 开发的关键步骤

接下来简单介绍一下开发 OpenGL 程序的几个关键步骤。

(1) 首先在当前 Activity 的 onCreate()方法中实例化 GLSurfaceView。

```
GLSurfaceView gl = new GLSurfaceView(this);
```

GLSurfaceView 是一个视图，继承至 SurfaceView，它内嵌的 surface 专门负责 OpenGL 渲染，用于构建一个使用 OpenGL ES 进行部分或全部渲染的应用程序。

(2) 自定义 MyRenderer 实现 android. opengl. GLSurfaceView. Renderer 接口，并重写三个抽象方法。

```
public abstract void onSurfaceCreated(GL10 gl, EGLConfig config)
public abstract void onSurfaceChanged(GL10 gl, int width, int height)
public abstract void onDrawFrame(GL10 gl)
```

这些方法很容易理解，onSurfaceCreated()方法在 surface 创建以后调用，onSurfaceChanged()方法是在 surface 发生改变以后调用，例如从竖屏切换到横屏的时候，onDrawFrame()方法是当任何时候调用一个画图方法的时候调用。

(3) 给 GLSurfaceView 对象注册一个 Renderer。

```
gl.setRenderer(new MyRenderer());
```

(4) 设置当前上下文内容视图。

```
this.setContentView(gl);
```

在 3D 图形编程中，需要注意以下两方面。

一是生命周期（Activity Life-cycle）。当 Activity 暂停和恢复时，必须通知 GLSurfaceView。当 Activity 暂停时需要通知 GLSurfaceView 调用 onPause()，当 Activity 恢复时需要通知 GLSurfaceView 调用 onResume()。这些调用允许 GLSurfaceView 暂停和恢复渲染的线程，同时允许 GLSurfaceView 释放和重构 OpenGL 显示。

二是渲染模式(Rendering Mode)。在设置 Renderer 后，可以通过调用 setRenderMode(int)来控制是否持续呈现或者点播。默认情况下持续呈现。

3. 3D 编程案例

下面通过三个 3D 应用案例，帮助读者初步了解 3D 编程的方法，体会一下 3D 应用的效果。

【案例 8.2】 设计一个自动旋转的非透明彩色立方体，立方体的各顶点颜色不同。

【说明】 本例需要至少设计两个类，一个是 Activity 类，在其中的 onCreate()方法内，需要创建一个 GLSurfaceView 对象，设置绘制模式。另一个是继承 Renderer 的子类，用于定义 3D 图形的绘制、渲染及其交互动作等。本例的开发重点是在逻辑代码上。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 3DCube_1 的 Android 项目。其应用程序名为 Cube_1，包名为 cn. com. sgmsc. Cube_1，Activity 组件名为 Cube1Activity。

(2) 设计布局。本例是设计一个自动旋转的立方体，屏幕上不需要其他的控件，所以在 res/layout 目录中的布局文件 main. xml 很简单，只需要定义一个线性布局即可。

(3) 开发启动 Activity 代码。打开 src/cn. com. sgmsc. Cube_1 包下的 Cube1Activity. java 文件,并编辑之。代码如下所示。

```
1  package cn.com.sgmsc.Cube1;
2
3  import android.app.Activity;
4  import android.opengl.GLSurfaceView;
5  import android.os.Bundle;
6  import android.view.WindowManager;
7  import android.view.Window;
8
9  public class Cube1Activity extends Activity {
10
11 GLSurfaceView GLview;
12
13 /** 当这个 activity 首次创建时调用 */
14 @Override
15 public void onCreate(Bundle savedInstanceState) {
16        super.onCreate(savedInstanceState);
17
18        /* 设置窗体为全屏模式,无标题 */
19        getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
20        getWindow().requestFeature(Window.FEATURE_NO_TITLE);
21
22        /* 创建一个 GLSurfaceView 用于绘制表面 */
23        GLSurfaceView GLview = new GLSurfaceView(this);
24
25        /* 设置 Renderer 用于执行实际的绘制工作 */
26        GLview.setRenderer(new MyGLRenderer(this));
27
28        /* 设置绘制模式为持续绘制 */
29        GLview.setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
30
31        /* 将创建好的 GLSurfaceView 设置为当前 Activity 的内容视图 */
32        setContentView(GLview);
33  }
34 }
```

第 23、26、32 行是 3D 开发不可缺少的步骤。其中,GLSurfaceView 对象 GLview 调用方法 setRenderer(new MyGLRenderer(this))对自动旋转的 3D 立方体进行绘制,定义绘制并自动旋转的代码将在自定义的 MyGLRenderer 类中实现。

(4) 开发 3D 效果渲染器代码。在 src/cn. com. sgmsc. Cube_1 包下创建一个代码文件 MyGLRenderer. java,该文件是定义一个 Renderer 子类,并重写其中的抽象方法 onDrawFrame()、onSurfaceChanged()和 onSurfaceCreated()。其代码如下所示。

```
package cn.com.sgmsc.Cube1;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView.Renderer;
import android.opengl.GLU;
```

```

public class MyGLRenderer implements Renderer {
 public MyGLRenderer(Cube1Activity main) {
      createBuffers();
 }

 public void onSurfaceCreated(GL10 gl, EGLConfig config) {
      gl.glDisable(GL10.GL_DITHER);              //颜色抖动据说可能严重影响性能,禁用
      gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);   //设置清除颜色缓冲区时用的 RGBA 颜色值

      gl.glEnable(GL10.GL_DEPTH_TEST);
      gl.glDepthFunc(GL10.GL_LEQUAL);
      gl.glClearDepthf(1f);
 }

 public void onSurfaceChanged(GL10 gl, int width, int height) {
      //宽高比
      float aspect = (float) width / (float) (height == 0 ? 1 : height);

      //设置视口
      gl.glViewport(0, 0, width, height);

      //设置当前矩阵堆栈为投影矩阵,并将矩阵重置为单位矩阵
      gl.glMatrixMode(GL10.GL_PROJECTION);
      gl.glLoadIdentity();

      GLU.gluPerspective(gl, 45.0f, aspect, 0.1f, 200.0f);
      GLU.gluLookAt(gl, 5f, 5f, 5f, 0f, 0f, 0f, 0, 1, 0);
 }
 public void onDrawFrame(GL10 gl) {
      //......省略了该方法的代码段,相应代码将在后面给出解析
 }

 private void createBuffers() {
      //创建顶点缓冲,顶点数组使用 float 类型,每个 float 长 4 个字节
      vertices = ByteBuffer.allocateDirect(data_vertices.length * 4);
      //设置字节顺序为本机顺序
      vertices.order(ByteOrder.nativeOrder());
      //通过一个 FloatBuffer 适配器,将 float 数组写入 ByteBuffer 中
      vertices.asFloatBuffer().put(data_vertices);
      //重置 Buffer 的当前位置
      vertices.position(0);

      //创建三角形顶点索引缓冲,索引使用 byte 类型,所以无须设置字节顺序,也无须写入适配
      triangles = ByteBuffer.allocateDirect(data_triangles.length * 2);
      triangles.put(data_triangles);
      triangles.position(0);

      colors = ByteBuffer.allocateDirect(data_colors.length * 4);
      colors.order(ByteOrder.nativeOrder());
      colors.asFloatBuffer().put(data_colors);
      colors.position(0);
 }

 private ByteBuffer vertices;
```

```
64 private ByteBuffer triangles;
65 private ByteBuffer colors;
66
67 private float anglex = 0f;
68 private float angley = 0f;
69 private float anglez = 0f;
70
71 private float[] data_vertices = {
72      1, 1, 1,   1, -1, 1,   -1, -1, 1,   -1, 1, 1,
73      1, 1, -1,   1, -1, -1,   -1, -1, -1,   -1, 1, -1,
74      };
75
76 private float[] data_colors = {
77      1, 1, 0, 1,   1, 0, 1, 1,   0, 1, 1, 1,   1, 0, 0, 1,
78      0, 0, 0, 1,   0, 0, 1, 1,   0, 1, 0, 1,   1, 1, 1, 1,
79      };
80
81 private byte[] data_triangles = {
82      0, 1, 2,   0, 2, 3,   0, 3, 7,   0, 7, 4,   0, 4, 5,   0, 5, 1,
83      6, 5, 4,   6, 4, 7,   6, 7, 3,   6, 3, 2,   6, 2, 1,   6, 1, 5
84      };
85 }
```

① 第 11～13 行，定义构造方法，在其中调用创建缓冲的方法 createBuffers()。createBuffers()方法在第 26～45 行中定义，在其中定义了顶点、三角形顶点索引和色彩三个缓冲区。

② 第 15～22 行是重写抽象方法 onSurfaceCreated()。

③ 第 24～37 行是重写抽象方法 onSurfaceChanged()。

④ 第 63～84 行声明和定义本类中使用的变量和数组，并赋初值。

接下来对省略的部分代码作解析。这部分省略的代码是重写抽象方法 onDrawFrame()。具体代码如下。

```
public void onDrawFrame(GL10 gl) {
     //清除颜色缓冲
     gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);

     //设置当前矩阵堆栈为模型堆栈,并重置堆栈
     //即随后的矩阵操作将应用到要绘制的模型上
     gl.glMatrixMode(GL10.GL_MODELVIEW);
     gl.glLoadIdentity();

     gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, new float[]{5, 5, 5, 1}, 0);

     //将旋转矩阵应用到当前矩阵堆栈上,即旋转模型
     gl.glRotatef(anglez, 0, 0, 1);
     gl.glRotatef(angley, 0, 1, 0);
     gl.glRotatef(anglex, 1, 0, 0);
     anglex += 0.1; //递增角度值以便每次以不同角度绘制
     angley += 0.2;
     anglez += 0.3;

     //启用顶点数组、法向量、颜色数组
     gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
```

```
22        gl.glEnableClientState(GL10.GL_NORMAL_ARRAY);
23        gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
24
25        //设置顶点数组指针为 ByteBuffer 对象 vertices
26        //第一个参数为每个顶点包含的数据长度(以第二个参数表示的数据类型为单位)
27        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertices);
28        gl.glColorPointer(4, GL10.GL_FLOAT, 0, colors);
29
30        //绘制 triangles 表示的三角形
31        gl.glDrawElements(GL10.GL_TRIANGLES, triangles.remaining(),
32                GL10.GL_UNSIGNED_BYTE, triangles);
33
34        //禁用顶点、法向量、颜色数组
35        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
36        gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
37 }
```

① 第 2～10 行是重置缓冲、堆栈和光源，为随后的绘制 3D 图形做准备。

② 第 12～18 行设置旋转模型。

③ 第 21～23 行启用相关数组，为 3D 图形提供数据。

④ 第 25～32 行绘制图形和着色。

⑤ 第 35、36 行禁用相关数组。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 3DCube_1 项目，便可在屏幕上看到一个八色渐变的彩色旋转正方体，如图 8-2 所示。

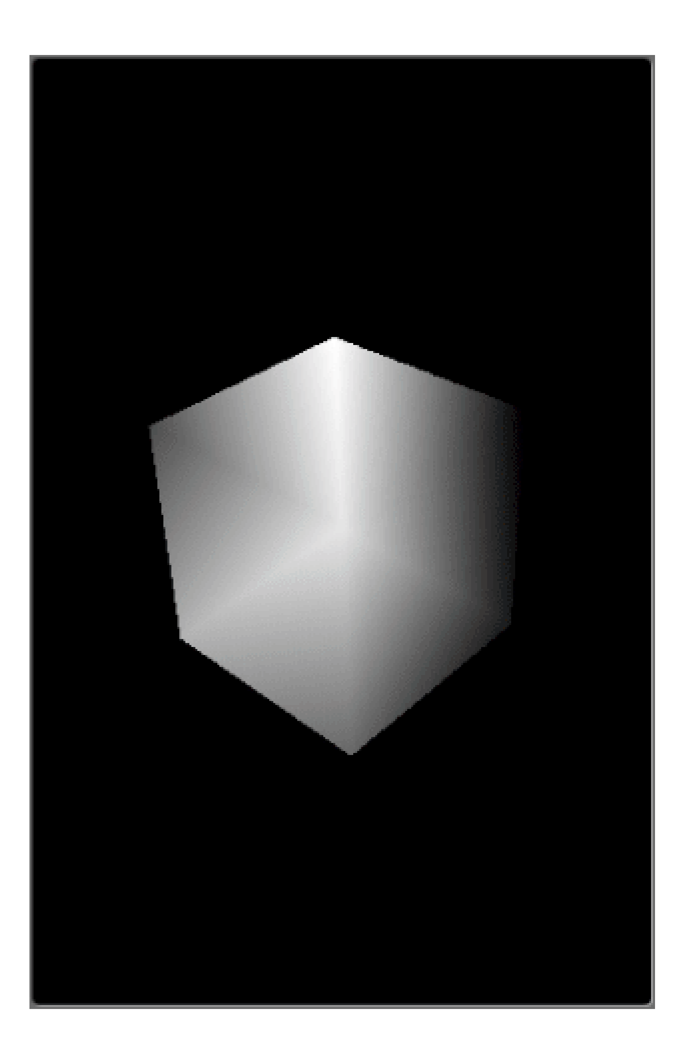


图 8-2　一个自动旋转的彩色立方体

【案例 8.3】 设计一个自动旋转的半透明的贴图立方体，立方体的每一面都贴着相同的图片，但各面的用光和前背景的色彩不一样。

【说明】 本例与例 8.2 的设计思想一样，不同之处在于定义 Renderer 子类时，增加了绘图纹理的处理和渲染，少了对顶点的颜色设置部分。本例的开发重点是在逻辑代码上。下面只对自定义的 Renderer 子类部分作较详细的解析。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 3DCube_2 的 Android 项目。其应用程序名为 Cube_2，包名为 cn.com.sgmsc.Cube_2，Activity 组件名为 Cube2Activity。

(2) 准备图片。准备一张方形图片作为纹理贴图，将其复制在 res/drawable-mdpi 目录中。注意：有些设备对图片的尺寸有要求，其长宽必须是 2^n，最大尺寸不能超过 256 或 1024 等。

(3) 设计布局。同样，res/layout 目录中的布局文件 main.xml 中也是只有简单的一个线性布局。

(4) 开发启动 Activity 代码。编辑 src/cn.com.sgmsc.Cube_2 包下的 Cube2Activity.java 文件，其代码与案例 8.2 中 Cube1Activity.java 的相同，在此不作赘述。

(5) 开发 3D 效果渲染器代码。在 src/cn.com.sgmsc.Cube_2 包下创建一个代码文件 MyGLRenderer.java，该文件是定义一个 Renderer 子类，并重写其中的 onDrawFrame()方法、

onSurfaceChanged()方法和 onSurfaceCreated()方法,3D 贴图立方体及动画效果在该子类中定义。其代码如下所示。

```
package cn.com.sgmsc.Cube2;

import java.io.InputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView.Renderer;
import android.opengl.GLUtils;
import android.opengl.GLU;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;

public class MyGLRenderer implements Renderer {

public MyGLRenderer(Context context) {
        this.context = context;
        createBuffers();
}

public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        //......省略一: 省略了该方法的代码段,相应代码将在后面给出解析
}

public void onSurfaceChanged(GL10 gl, int width, int height) {
        //......省略二: 省略了该方法的代码段,相应代码将在后面给出解析
}

public void onDrawFrame(GL10 gl) {
        //......省略三: 省略了该方法的代码段,相应代码将在后面给出解析
}

        private void createBuffers() {
        //创建顶点缓冲,顶点数组使用 float 类型,每个 float 长 4 个字节
        vertices = ByteBuffer.allocateDirect(data_vertices.length * 4);
        //设置字节顺序为本机顺序
        vertices.order(ByteOrder.nativeOrder());
        //通过一个 FloatBuffer 适配器,将 float 数组写入 ByteBuffer 中
        vertices.asFloatBuffer().put(data_vertices);
        //重置 Buffer 的当前位置
        vertices.position(0);

        //创建索引缓冲,索引使用 byte 类型,所以无须设置字节顺序,也无须写入适配
        triangles = ByteBuffer.allocateDirect(data_triangles.length * 2);
        triangles.put(data_triangles);
        triangles.position(0);

        tvertices = ByteBuffer.allocateDirect(data_tvertices.length * 4);
        tvertices.order(ByteOrder.nativeOrder());
        tvertices.asFloatBuffer().put(data_tvertices);
```

```
        tvertices.position(0);
}

private void loadTexture(GL10 gl) {
        //......省略四：省略了该方法的代码段，相应代码将在后面给出解析
}

private Context context;
private ByteBuffer vertices;
private ByteBuffer triangles;
private ByteBuffer tvertices;
private int texture;

rivate float anglex = 0f;
private float angley = 0f;
private float anglez = 0f;

private float[] data_vertices = {
        -5.0f, -5.0f, -5.0f, -5.0f, 5.0f, -5.0f, 5.0f, 5.0f, -5.0f, 5.0f, 5.0f, -5.0f,
        5.0f, -5.0f, -5.0f, -5.0f, -5.0f, -5.0f,
        -5.0f, -5.0f, 5.0f, 5.0f, -5.0f, 5.0f, 5.0f, 5.0f, 5.0f, 5.0f, 5.0f, 5.0f, -5.0f,
        5.0f, 5.0f, -5.0f, -5.0f, 5.0f,
        -5.0f, -5.0f, -5.0f, 5.0f, -5.0f, -5.0f, 5.0f, -5.0f, 5.0f, 5.0f, -5.0f, 5.0f,
        -5.0f, -5.0f, 5.0f, -5.0f, -5.0f, -5.0f,
        5.0f, -5.0f, -5.0f, 5.0f, 5.0f, -5.0f, 5.0f, 5.0f, 5.0f, 5.0f, 5.0f, 5.0f, 5.0f,
        -5.0f, 5.0f, 5.0f, -5.0f, -5.0f,
        5.0f, 5.0f, -5.0f, -5.0f, 5.0f, -5.0f, -5.0f, 5.0f, 5.0f, -5.0f, 5.0f, 5.0f,
        5.0f, 5.0f, 5.0f, 5.0f, 5.0f, -5.0f,
        -5.0f, 5.0f, -5.0f, -5.0f, -5.0f, -5.0f, -5.0f, -5.0f, 5.0f, -5.0f, -5.0f,
        5.0f, -5.0f, 5.0f, 5.0f, -5.0f, 5.0f, -5.0f,
        };

private byte[] data_triangles = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, };

//设置纹理坐标
private float[] data_tvertices = {
        1.0000f, 1.0000f, 1.0000f, 0.0000f, 0.0000f, 0.0000f, 0.0000f, 0.0000f, 0.0000f,
        1.0000f, 1.0000f, 1.0000f,
        0.0000f, 1.0000f, 1.0000f, 1.0000f, 1.0000f, 0.0000f, 1.0000f, 0.0000f, 0.0000f,
        0.0000f, 0.0000f, 1.0000f,
        0.0000f, 1.0000f, 1.0000f, 1.0000f, 1.0000f, 0.0000f, 1.0000f, 0.0000f, 0.0000f,
        0.0000f, 0.0000f, 1.0000f,
        0.0000f, 1.0000f, 1.0000f, 1.0000f, 1.0000f, 0.0000f, 1.0000f, 0.0000f, 0.0000f,
        0.0000f, 0.0000f, 1.0000f,
        0.0000f, 1.0000f, 1.0000f, 1.0000f, 1.0000f, 0.0000f, 1.0000f, 0.0000f, 0.0000f,
        0.0000f, 0.0000f, 1.0000f,
        0.0000f, 1.0000f, 1.0000f, 1.0000f, 1.0000f, 0.0000f, 1.0000f, 0.0000f, 0.0000f,
        0.0000f, 0.0000f, 1.0000f,
        };
}
```

本例是设计一个3D贴图立方体，所以与案例8.2中的MyGLRenderer子类定义相比，这里多了加载贴图纹理的方法loadTexture()。

接下来对省略的部分代码作解析。

省略一：这部分省略的代码是重写抽象方法 onSurfaceCreated()。具体代码如下。

```
1  public void onSurfaceCreated(GL10 gl, EGLConfig config) {
2
3      boolean SEE_THRU = true;                    //为设置透明效果的判断值
4      gl.glDisable(GL10.GL_DITHER);               //禁用颜色抖动,因为可能严重影响性能
5      gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);    //设置清除颜色缓冲区时用的 RGBA 颜色值
6
7
8      //在位置(1,1,1)处定义光源
9      float lightAmbient[] = new float[] { 0.3f, 0.3f, 0.3f, 1 };
10     float lightDiffuse[] = new float[] { 1, 1, 1, 1 };
11     float[] lightPos = new float[] { 1, 1, 1, 1 };
12     gl.glEnable(GL10.GL_LIGHTING);
13     gl.glEnable(GL10.GL_LIGHT0);
14     //设置环境光
15     gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient, 0);
16     //设置漫射光
17     gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);
18     //设置光源位置
19     gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 0);
20
21     //定义立方体材质
22     float matAmbient[] = new float[] { 1, 1, 1, 1 };
23     float matDiffuse[] = new float[] { 1, 1, 1, 1 };
24     gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, matAmbient, 0);
25     gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, matDiffuse, 0);
26
27     //设置需要的 OpenGL 项
28     gl.glEnable(GL10.GL_DEPTH_TEST);
29     gl.glDepthFunc(GL10.GL_LEQUAL);
30     gl.glClearDepthf(1f);
31     gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
32
33     //设置透明效果
34     if (SEE_THRU) {
35         gl.glDisable(GL10.GL_DEPTH_TEST);
36         gl.glEnable(GL10.GL_BLEND);
37         gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);
38     }
39
40     //加载纹理贴图
41     gl.glEnable(GL10.GL_TEXTURE_2D);
42     loadTexture(gl);
43 }
```

① 第 9～19 行定义用光的属性,包括光源、环境光、漫射光。

② 第 22～25 行定义立方体的材质。

③ 第 34～38 行定义透明效果。

④ 第 42 行加载纹理图片。

省略二：这部分省略的代码是重写抽象方法 onSurfaceChanged()。具体代码如下。

```
public void onSurfaceChanged(GL10 gl, int width, int height) {

```

```
3        //设置视口
4        gl.glViewport(0, 0, width, height);
5
6        //设置当前矩阵堆栈为投影矩阵,并将矩阵重置为单位矩阵
7        gl.glMatrixMode(GL10.GL_PROJECTION);
8        gl.glLoadIdentity();
9
10       //宽高比
11       float aspect = (float) width / (float) (height == 0 ? 1 : height);
12       GLU.gluPerspective(gl, 45.0f, aspect, 0.1f, 200.0f);
13       //设置投影变换矩阵,显示稍大些的立方体
14       GLU.gluLookAt(gl, -20f, -20f, 20f, 0f, 0f, 0f, 0, 0, 1);
15 }
```

① 第 12 行,定义一个透视投影变换。gluPerspective()是 Android OpenGL ES 的 GLU 包内的一个定义透视投影变换的方法,其格式为:

```
GLU.gluPerspective(GL10 gl, float fovy, float aspect ,float zNear, float zFar)
```

其中,fovy 定义视锥的 view angle;aspect 定义视锥的宽高比;zNear 定义裁剪面的近间隔;zFar 定义创建面的远间隔。

② 第 14 行,设置 modelview 变换矩阵。gluLookAt()是 Android OpenGL ES 的 GLU 包内的一个直观的设置 modelview 变换矩阵的方法,其格式为:

```
GLU.gluLookAt(GL10 gl, float eyeX, float eyeY, float eyeZ, float centerX, float centerY, float
centerZ, float upX, float upY, float upZ)
```

其中,eyeX,eyeY,eyeZ 指定观测点的空间坐标;centerX,centerY,centerZ 指定被观测物体的参考点的坐标;upX,upY,upZ 指定观测点标的目标为"上"的向量。

省略三:这部分省略的代码是重写抽象方法 onDrawFrame ()。具体代码如下。

```
public void onDrawFrame(GL10 gl) {
    //清除颜色缓冲
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);

    //设置当前矩阵堆栈为模型堆栈,并重置堆栈
    //即随后的矩阵操作将应用到要绘制的模型上
    gl.glMatrixMode(GL10.GL_MODELVIEW);
    gl.glLoadIdentity();
    gl.glTranslatef(0, 0, -3.0f);

    //将旋转矩阵应用到当前矩阵堆栈上,即旋转模型
    gl.glRotatef(anglez, 0, 0, 1);
    gl.glRotatef(angley, 0, 1, 0);
    gl.glRotatef(anglex, 1, 0, 0);
    anglex += 0.1; //递增角度值以便每次以不同角度绘制
    angley += 0.2;
    anglez += 0.3;

    //启用顶点数组、法向量、纹理坐标数组
    gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
    gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);

    //设置正面
```

```
24        gl.glFrontFace(GL10.GL_CW);
25
26        //设置顶点数组指针为 ByteBuffer 对象 vertices
27        //第一个参数为每个顶点包含的数据长度(以第二个参数表示的数据类型为单位)
28        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertices);
29        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, tvertices);
30        //绑定纹理
31        gl.glBindTexture(GL10.GL_TEXTURE_2D, texture);
32
33        //绘制 triangles 表示的三角形
34        gl.glDrawElements(GL10.GL_TRIANGLES, triangles.remaining(),
35                GL10.GL_UNSIGNED_BYTE, triangles);
36
37        //禁用顶点、法向量、纹理坐标数组
38        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
39        gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
40 }
```

① 第 8 行，将当前矩阵设置为单位矩阵。

② 第 9 行，设置移动。这里的 glTranslatef(0, 0, −3.0f);表示将视图向 Z 轴负向平移三个单位。

省略四：这部分省略的代码是定义对物体贴纹理图片的方法 loadTexture()。具体代码如下。

```
private void loadTexture(GL10 gl) {
    InputStream bitmapStream = null;
    Bitmap bitmap = null;
    try {
        bitmapStream = context.getResources().openRawResource(R.drawable.tree);
                                                //从图资源中取图片
        bitmap = BitmapFactory.decodeStream(bitmapStream);
                                                //从流中加载并解码图片生成 Bitmap 对象

        int[] textures = new int[1];
        gl.glGenTextures(1, textures, 0);   //生成一组纹理并把纹理的 ID 存入数组参数中
        texture = textures[0];

        gl.glBindTexture(GL10.GL_TEXTURE_2D, texture);
                                                //将指定 ID 的纹理绑定到指定的目标中去

        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_NEAREST);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_LINEAR);

        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_S, GL10.GL_REPEAT);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_T, GL10.GL_REPEAT);

        GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmap, 0);
                                                //将 Bitmap 对象设置到纹理目录中

    } finally {
        if (bitmap != null)
          bitmap.recycle();

        if (bitmapStream != null)
```

```
27              try {
28                      bitmapStream.close();
29              } catch (IOException e) {
30              }
31      }
32 }
```

第 14～18 行设置纹理参数，这里设置了 4 个参数：GL_TEXTURE_MIN_FILTER 和 GL_TEXTURE_MAG_FILTER 指定纹理在被缩小或放大时使用的过滤方式，GL_LINEAR（线性插值）要比 GL_NEAREST（最近点）效果好，但前者需要更多的运算开销；GL_TEXTURE_WRAP_S 和 GL_TEXTURE_WRAP_T 表示当贴图坐标不在 0.0～1.0 之间时如何处理，这里使用 GL_REPEAT，即平铺贴图。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 3DCube_2 项目，便可在屏幕上看到一个自动顺时针旋转的贴图正方体，如图 8-3 所示。



图 8-3　一个自动顺时针旋转的贴图正方体

【案例 8.4】 设计一个立体多面体，使用触控笔可以旋转这个 3D 图形。

【说明】 本例在 3D 图形上增加了触摸功能，需要在 GLSurfaceView 对象上添加一个 onTouchEvent()的回调方法。本例的重点工作仍在开发逻辑代码上。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 3DGraphics 的 Android 项目。其应用程序名为 3DTouch，包名为 cn.com.sgmsc.OpenGL，Activity 组件名为 OpenGLActivity。

(2) 开发逻辑代码。打开 src/cn.com.sgmsc.OpenGL 包下的 OpenGLActivity.java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.OpenGL;

import android.app.Activity;
import android.os.Bundle;

public class OpenGLActivity extends Activity {
MyGLSurfaceView myGLSurfaceView;                          //自定义 GLSurfaceView
   @Override
   public void onCreate(Bundle savedInstanceState) {      //Activity 创建时被调用
       super.onCreate(savedInstanceState);
       myGLSurfaceView = new MyGLSurfaceView(this);       //创建一个自定义的 GLSurfaceView
       this.setContentView(myGLSurfaceView);              //设置当前的用户界面
       myGLSurfaceView.requestFocus();                    //获得焦点
       myGLSurfaceView.setFocusableInTouchMode(true);//设置可以触控
   }
   @Override
   protected void onResume(){
       super.onResume();
       if(myGLSurfaceView != null){                       //当 myGLSurfaceView 不为空时
            myGLSurfaceView.onResume();
```

```
21          }
22      }
23      @Override
24      protected void onPause(){
25          super.onPause();
26          if(myGLSurfaceView != null){              //当 myGLSurfaceView 不为空时
27              myGLSurfaceView.onPause();
28          }
29      }
30  }
```

第 17～22 行，重写了 onResume()方法，第 24～29 行重写了 onPause()方法。这样做的目的是当该应用从前台转到后台，再从后台变为前台后，屏幕上的 3D 图形能保持先前的状态。

（3）开发 3D 效果渲染器及触摸操作代码。在 src/cn.com.sgmsc.OpenGL 包下创建一个代码文件 MyGLSurfaceView.java，该文件是定义一个 GLSurfaceView 子类，在这里定义渲染器 MyRenderer，它继承自 GLSurfaceView.Renderer。其代码如下所示。

```
package cn.com.sgmsc.OpenGL;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.IntBuffer;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.content.Context;
import android.opengl.GLSurfaceView;
import android.view.MotionEvent;

public class MyGLSurfaceView extends GLSurfaceView{
    MyRenderer myRenderer;                                   //自定义的渲染器
    private IntBuffer   mVertexBuffer;                       //顶点坐标数据缓冲
    private IntBuffer   mColorBuffer;                        //顶点着色数据缓冲
    private final float TOUCH_SCALE_FACTOR = 180.0f/320;//角度缩放比例
    private float mPreviousY;                                //上次的触控位置 Y 坐标
    private float mPreviousX;                                //上次的触控位置 X 坐标
    float yAngle = 0;                                        //绕 Y 轴旋转的角度
    float zAngle = 0;                                        //绕 Z 轴旋转的角度
    int vertexCount;                                         //顶点的个数
public MyGLSurfaceView(Context context){                     //构造器
     super(context);
     myRenderer  =  new MyRenderer();                        //创建渲染器
     this.setRenderer(myRenderer);                           //设置渲染器
     this.setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);          //设置渲染模式
     this.initVertexBuffer();                                //初始化顶点坐标数组
     this.initColorBuffer();                                 //初始化颜色数组
}
 @Override
 public boolean onTouchEvent(MotionEvent e){                 //触摸事件的回调方法
     float x  =  e.getX();                                   //得到 X 坐标
     float y  =  e.getY();                                   //得到 Y 坐标
     switch (e.getAction()) {
        case MotionEvent.ACTION_MOVE:                        //触控笔移动
             float dy  =  y  -  mPreviousY;                  //计算触控笔 Y 位移
```

```
                float dx = x - mPreviousX;              //计算触控笔 X 位移
                yAngle += dx * TOUCH_SCALE_FACTOR;      //设置沿 Y 轴旋转角度
                zAngle += dy * TOUCH_SCALE_FACTOR;      //设置沿 Z 轴旋转角度
                requestRender();                        //重绘画面
            }
            mPreviousY = y;                             //记录触控笔位置
            mPreviousX = x;                             //记录触控笔位置
            return true;//返回 true
        }
private void initVertexBuffer(){                        //初始化顶点坐标数据
      //......省略一：省略了该方法的代码段,相应代码将在后面给出解析
}
private void initColorBuffer(){//初始化颜色数组
      //......省略二：省略了该方法的代码段,相应代码将在后面给出解析
}
//自定义的渲染器
private class MyRenderer implements GLSurfaceView.Renderer{
      @Override
      public void onDrawFrame(GL10 gl) {                //绘制方法
            //清除颜色缓存于深度缓存
              gl.glClear(GL10.GL_COLOR_BUFFER_BIT|GL10.GL_DEPTH_BUFFER_BIT);
              gl.glMatrixMode(GL10.GL_MODELVIEW);       //设置当前矩阵为模式矩阵
              gl.glLoadIdentity(); //设置当前矩阵为单位矩阵
              gl.glTranslatef(0, 0f, -3f);
              gl.glRotatef(yAngle, 0, 1, 0);            //沿 Y 轴旋转
              gl.glRotatef(zAngle, 0, 0, 1);            //沿 X 轴旋转
              gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);   //启用顶点坐标数组
              gl.glEnableClientState(GL10.GL_COLOR_ARRAY);    //启用顶点颜色数组
              gl.glVertexPointer(//为画笔指定顶点坐标数据
                          3,//每个顶点的坐标数量为 3  xyz
                          GL10.GL_FIXED,                //顶点坐标值的类型为 GL_FIXED
                          0, mVertexBuffer              //顶点坐标数据
              );
              gl.glColorPointer(                        //为画笔指定顶点着色数据
                          4, //设置颜色的组成成分,必须为 4—RGBA
                          GL10.GL_FIXED,                //顶点颜色值的类型为 GL_FIXED
                          0, mColorBuffer               //顶点着色数据
              );
              gl.glDrawArrays(GL10.GL_TRIANGLES, 0, vertexCount);          //绘制图形
      }
      @Override
        public void onSurfaceChanged(GL10 gl, int width, int height) {
              gl.glViewport(0, 0, width, height);       //设置视窗大小及位置
              gl.glMatrixMode(GL10.GL_PROJECTION);      //设置当前矩阵为投影矩阵
              gl.glLoadIdentity();                      //设置当前矩阵为单位矩阵
              float ratio = (float)width/height;        //计算透视投影的比例
              gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10); //调用此方法计算产生透视投影矩阵
        }
      @Override
        public void onSurfaceCreated(GL10 gl, EGLConfig config) {          //创建时被调用
              gl.glDisable(GL10.GL_DITHER);             //关闭抗抖动
              gl.glEnable(GL10.GL_DEPTH_TEST);          //启用深度测试
        }
  }
}
```

第 30～45 行，重写了 onTouchEvent()方法。在该方法中定义：当触摸笔上下移动时，物体沿 Z 轴旋转，当左右移动时，物体沿 X 轴旋转。

接下来对省略的部分代码作解析。

省略一：这部分省略的代码是初始化顶点坐标数据方法 initVertexBuffer()。具体代码如下。

```
1  private void initVertexBuffer(){                //初始化顶点坐标数据
2      vertexCount = 15;                           //顶点的个数
3      final int U = 10000;                        //创建像素单位
4      int vertices[] = new int[]{
5          0,0,0,
6          -6*U,-6*U,0,
7          -6*U,6*U,0,
8          0,0,0,
9          -6*U,6*U,0,
10         0,12*U,0,
11         0,0,0,
12         0,12*U,0,
13         6*U,6*U,0,
14         0,0,0,
15         6*U,6*U,0,
16         6*U,-6*U,0,
17         0,0,0,
18         6*U,-6*U,0,
19         -6*U,-6*U,0
20     };
21     //创建顶点坐标数据缓冲
22     ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
23     vbb.order(ByteOrder.nativeOrder());         //设置字节顺序
24     mVertexBuffer = vbb.asIntBuffer();          //转换为 int 型缓冲
25     mVertexBuffer.put(vertices);                //向缓冲区中放入顶点坐标数据
26     mVertexBuffer.position(0);                  //设置缓冲区起始位置
27 }
```

第 5、8、11、14、17 行定义 5 个顶点的坐标，它们重合在坐标原点上。

省略二：这部分省略的代码是初始化颜色数组方法 initColorBuffer()。具体代码如下。

```
private void initColorBuffer(){                 //初始化颜色数组
    final int one = 65535;
    int colors[] = new int[]{                   //顶点颜色值数组，每个顶点 4 个色彩值 RGBA
        one,one,0,0,                            //黄色
        0,0,one,0,                              //蓝色
        0,0,one,0,
        one,one,0,0,
        0,0,one,0,
        0,0,one,0,
        one,one,0,0,
        0,0,one,0,
        0,0,one,0,
        one,one,0,0,
        0,0,one,0,
        0,0,one,0,
        one,one,0,0,
        0,0,one,0,
```

```
18             0,0,one,0,
19         };
20         //创建顶点着色数据缓冲
21         ByteBuffer cbb = ByteBuffer.allocateDirect(colors.length * 4);
22         cbb.order(ByteOrder.nativeOrder());        //设置字节顺序
23         mColorBuffer = cbb.asIntBuffer();          //转换为 int 型缓冲
24         mColorBuffer.put(colors);                  //向缓冲区中放入顶点着色数据
25         mColorBuffer.position(0);                  //设置缓冲区起始位置
26 }
```

第 4、7、10、13、16 行定义 5 个顶点的颜色为黄色，其余各点的颜色为蓝色。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 3DGraphics 项目，初始进入界面如图 8-4 所示。当拖动了 3D 图形之后，可看到经过旋转变形的图形，如图 8-5 所示。

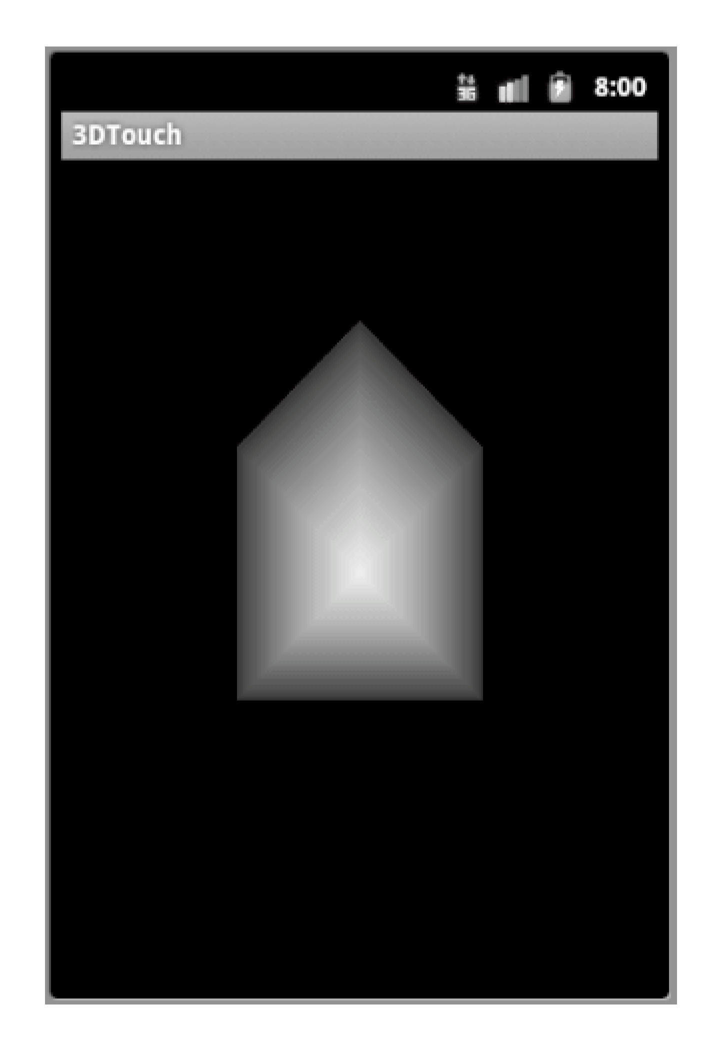


图 8-4　初始进入程序时的图形

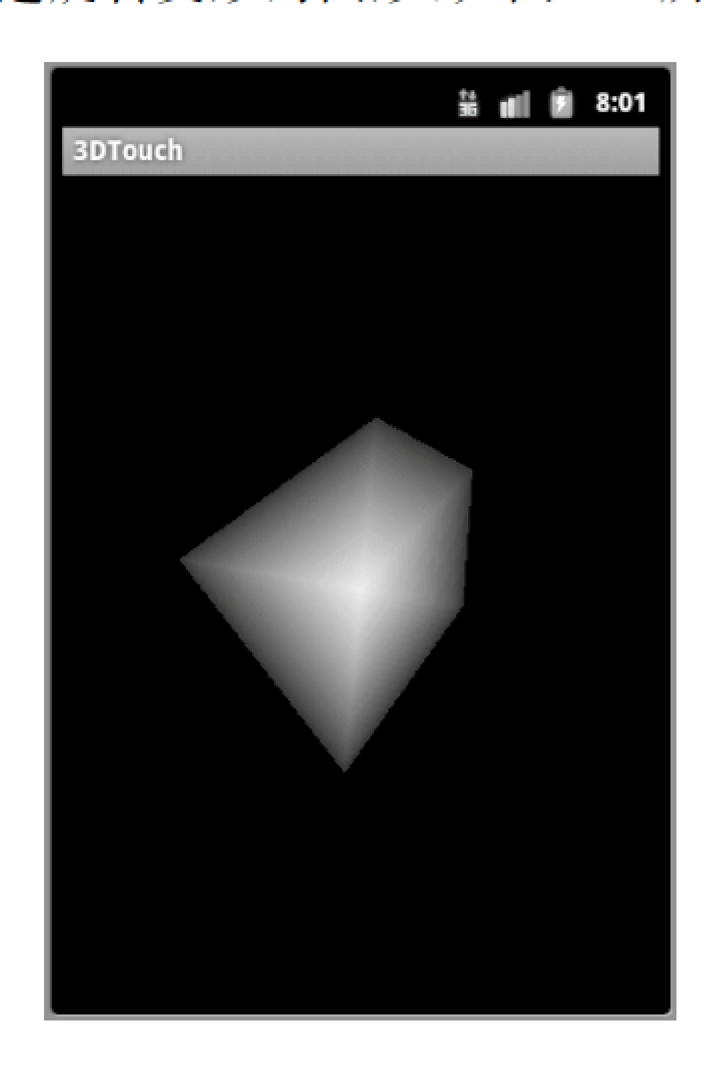


图 8-5　经过拖动旋转后的图形

8.2　动画播放

在开发用户界面时，除了使用控件布局，或使用 2D、3D 图形外，还可以使用动画播放来提高用户的体验。本节将介绍两种动画播放技术：帧动画（Frame Animation）和补间动画（Tween Animation）。

在 Android 中定义动画，既可以使用 XML 文件进行描述，也可以使用 Java 代码编程。如果使用 XML 文件描述动画，其 XML 文件存放在 res/anim 目录下，这个 anim 目录需要用户创建。本书推荐使用 XML 文件定义动画，因为使用 XML 文件的可读性较高，程序开发的效率也较高。

下面分别来介绍两个动画技术及其使用方法。

8.2.1　帧动画

1. 帧动画简介

帧动画是连续地播放一系列的图片文件，形成动画。它是比较传统的动画方式，形成动画

的主要要素有三个：多幅图片、播放顺序和持续时间。

帧动画用到类 AnimationDrawable，这个类包含在 android. graphics. drawable 包下。每个帧动画都是一个 AnimationDrawable 对象。

如果在 XML 文件中定义动画，其常用的属性如表 8-5 所示。

表 8-5 使用 XML 描述帧动画的常用属性说明

标签名	属　　性	说　　明
<animation-list>	android:oneshot：如果设置为 true，则该动画只播放一次，然后停止在最后一帧	Frame Animation 的根标签，包含若干个<item>
<item>	android:drawable：图片帧的引用 android:duration：图片帧的停留时间 android:visible：图片帧是否可见	每个<item>标记定义一个图片帧，其中包含图片资源的引用等属性

2. 帧动画的使用

使用帧动画进行开发需要注意的是，如果程序一启动就播放动画，不能在 onCreate()方法中调用 start()方法，而要在 onWindowFocusChanged()方法中调用 start()方法。下面通过一个案例来说明帧动画的用法。该案例是通过按钮的单击来启动一个动画的。

【案例 8.5】 使用 Frame Animation 设计一个能循环播放小狗图片的动画，该动画由单击按钮启动。

【说明】 按照帧动画的三要素，首先需要准备一组图片，并且最好这些图片的尺寸一致。

【开发步骤及解析】

(1) 准备素材。收集并整理一套图片，将其尺寸裁剪一致，按照 p01. png，p02. png…的规则命名。

(2) 创建项目。在 Eclipse 中创建一个名为 FrameAnimation 的 Android 项目。其应用程序名为 FAnimation，包名为 cn. com. sgmsc. FAnimation，Activity 组件名为 FAnimationActivity。

(3) 准备图片。将准备好的图片复制到 res/drawable-mdpi 目录中。

(4) 定义动画。在 res 目录下新建一个 anim 目录，并在该目录下创建一个名为 frame_ani. xml 文件并编写动画的描述代码，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<animation - list xmlns:android = "http://schemas.android.com/apk/res/android"
      android:oneshot = "false">
    < item android:drawable = "@drawable/p01"
         android:duration = "1000"
         android:visible = "true"/>
    < item android:drawable = "@drawable/p02"
         android:duration = "1000"
         android:visible = "true"/>
    < item android:drawable = "@drawable/p03"
          android:duration = "1000"
          android:visible = "true"/>
     < item android:drawable = "@drawable/p04"
          android:duration = "1000"
          android:visible = "true"/>
     < item android:drawable = "@drawable/p05"
```

```
17            android:duration = "1000"
18            android:visible = "true"/>
19        < item android:drawable = "@drawable/p06"
20            android:duration = "1000"
21            android:visible = "true"/>
22        < item android:drawable = "@drawable/p07"
23            android:duration = "1000"
24            android:visible = "true"/>
25        < item android:drawable = "@drawable/p08"
26            android:duration = "1000"
27            android:visible = "true"/>
28        < item android:drawable = "@drawable/p09"
29            android:duration = "1000"
30            android:visible = "true"/>
31        < item android:drawable = "@drawable/p10"
32            android:duration = "1000"
33            android:visible = "true"/>
34        < item android:drawable = "@drawable/p11"
35            android:duration = "1000"
36            android:visible = "true"/>
37        < item android:drawable = "@drawable/p12"
38            android:duration = "1000"
39            android:visible = "true"/>
40        < item android:drawable = "@drawable/p13"
41            android:duration = "1000"
42            android:visible = "true"/>
43        < item android:drawable = "@drawable/p14"
44            android:duration = "1000"
45            android:visible = "true"/>
46        < item android:drawable = "@drawable/p15"
47            android:duration = "1000"
48            android:visible = "true"/>
49        < item android:drawable = "@drawable/p16"
50            android:duration = "1000"
51            android:visible = "true"/>
52        < item android:drawable = "@drawable/p17"
53            android:duration = "1000"
54            android:visible = "true"/>
55        < item android:drawable = "@drawable/p18"
56            android:duration = "1000"
57            android:visible = "true"/>
58    </animation - list >
```

① 第 3 行设置动画是循环播放。

② 每一个< item >标签里设置一幅图的播放属性，包括指定图片的文件名，指定图片播放时间是停留 1s，图片可见。

(5) 设计布局。编写 res/layout 目录下的 main. xml 文件，其代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
      android:orientation = "vertical"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent"
      >                                          <! -- 声明一个垂直分布的线性布局 -->
< ImageView
```

```
8          android:id = "@ + id/iv"
9          android:background = "@anim/frame_ani"
10         android:layout_width = "wrap_content"
11         android:layout_height = "wrap_content"
12         android:layout_gravity = "center_horizontal"
13         />                                        <! -- 声明一个 ImageView 对象 -->
14 <Button
15         android:id = "@ + id/btn"
16         android:text = "开始动画"
17         android:layout_width = "fill_parent"
18         android:layout_height = "wrap_content"
19         android:layout_gravity = "center_horizontal"
20         />                                        <! -- 声明一个 Button 对象 -->
21 </LinearLayout>
```

第 9 行设置该 ImageView 控件的图片源为帧动画 XML 文件。

(6) 开发逻辑代码。打开并编辑 src/cn. com. sgmsc. Fanimation 包下的文件 FanimationActivity. java,代码如下所示。

```
1  package cn.com.sgmsc.FAnimation;
2
3  import android.app.Activity;                              //引入相关类
4  import android.graphics.drawable.AnimationDrawable;       //引入相关类
5  import android.os.Bundle;                                 //引入相关类
6  import android.view.View;                                 //引入相关类
7  import android.view.View.OnClickListener;                 //引入相关类
8  import android.widget.Button;                             //引入相关类
9  import android.widget.ImageView;                          //引入相关类
10
11 public class FAnimationActivity extends Activity {
12     @Override
13     public void onCreate(Bundle savedInstanceState) {     //重写 onCreate()方法
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.main);
16         Button btn = (Button)findViewById(R.id.btn);
17         btn.setOnClickListener(new OnClickListener() {    //为按钮设置监听器
18           @Override
19           public void onClick(View v) {                    //重写 onClick()方法
20              ImageView iv = (ImageView)findViewById(R.id.iv);
21              iv.setBackgroundResource(R.anim.frame_ani);
22              AnimationDrawable ad = (AnimationDrawable)iv.getBackground();
23              ad.start();                                   //启动 AnimationDrawable
24           }
25         });
26     }
27 }
```

① 第 21 行设置 iv 对象的图片源来自于 drawable-mdpi 目录中的 frame_ani. xml。

② 第 22 行创建一个 AnimationDrawable 对象 ad,第 23 行使用 start()方法启动 ad。注意,在 onCreate()方法内不能直接使用 ad. start()。这里的 start()方法是在按钮的 OnClickListener 监听内的 onClick()方法中调用的。

【运行结果】 在 Eclipse 中启动 Android 模拟器,然后运行 FrameAnimation 项目。单击“开始动画”按钮后,在屏幕上循环播放一系列的图片,如图 8-6 所示。







图 8-6　播放帧动画的效果图

8.2.2　补间动画

1. 补间动画简介

补间动画是通过一系列的指令，将一个 View 对象进行位置、尺寸、旋转等变换，形成动画。这个 View 对象可以是一幅图片，也可以是其他的文本、按钮等可视对象。补间动画涉及的类有 Animation、AnimationSet，这些类都位于 android. view. animation 包下。

补间动画也可以在 res/anim 目录中的 XML 文件里进行声明定义。在 XML 文件中指定动画使用何种变换、何时进行变换及动画持续多长时间等。补间动画常用的标签及属性如表 8-6 所示。

表 8-6　使用 XML 描述补间动画的常用属性说明

标签名	属　　性		说　　明
<set>	shareInterpolator：在子元素中共享插入器		包含其他动画变换的容器
<alpha>	fromAlpha：起始透明度	toAlpha：终止透明度	取值为 0.0～1.0，其中 0.0 为全透明
<scale>	fromXScale：X 的起始值	toXScale：X 的终止值	实现尺寸变换，其中 1.0 为原始大小
	fromYScale：Y 的起始值	toYScale：Y 的终止值	
	pivotX：中心的 X 坐标	pivotY：中心的 Y 坐标	
<translate>	fromXDelta：X 起始位置	toXDelta：X 终止位置	实现水平或垂直移动，以“%”结尾代表相对于自身的比例；以“%p”结尾代表相对于父控件的比例；不以任何后缀结尾代表绝对值
	fromYDelta：Y 起始位置	toYDelta：Y 终止位置	
<rotate>	fromDegree：开始位置	toDegree：结束位置	实现旋转，可以指定旋转定位点
	pivotX：中心的 X 坐标	pivotY：中心的 Y 坐标	
<Interpolator>	无		插入器，描述变换的速度曲线

表 8-6 中列出的是各标签特有的属性，在这些标签中有一些共有的属性，如表 8-7 所示。

表 8-7　使用 XML 描述补间动画标签的常用公共属性说明

属　　性	说　　明
duration	变换持续的时间，以毫秒为单位
startOffset	变换开始的时间，以毫秒为单位
repeatCount	定义该动画重复的次数
interpolator	为每个子标记变换设置插入器，系统已经设置好一些插入器，可以在 R. anim 包下找到

2. 补间动画的使用

在使用XML定义补间动画时，所有的变换要放在一组<set></set>标签之间。一组<set></set>标签内可以定义多个变换，甚至可以是另一组<set></set>标签。下面通过一个案例来说明补间动画的用法。

【案例8.6】 使用Tween Animation设计一个从暗到亮、逐渐放大、绕中心旋转的Android小机器人动画，并且可以动态指定ImageView中的图片源。

【说明】 在程序中使用两个按钮为ImageView控件动态指定图片源。

【开发步骤及解析】

(1) 创建项目。在Eclipse中创建一个名为TweenAnimation的Android项目。其应用程序名为TAnimation，包名为cn.com.sgmsc.TAni，Activity组件名为TAnimationActivity。

(2) 准备图片。将准备好的图片复制到res/drawable-mdpi目录中。

(3) 定义动画。在res目录下新建一个anim目录，并在该目录下创建一个名为tween_ani.xml的文件，并编写动画的描述代码，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 透明度的变换 -->
    <alpha
        android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="5000"
    />
    <!-- 尺寸的变换 -->
    <scale
        android:interpolator= "@android:anim/accelerate_decelerate_interpolator"
        android:fromXScale="0.0"
        android:toXScale="1.0"
        android:fromYScale="0.0"
        android:toYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fillAfter="false"
        android:duration="9000"
    />
    <!-- 位置的变换 -->
    <translate
        android:fromXDelta="30"
        android:toXDelta="0"
        android:fromYDelta="30"
        android:toYDelta="0"
        android:duration="10000"
    />
    <!-- 旋转变换 -->
    <rotate
        android:interpolator="@android:anim/accelerate_decelerate_interpolator"
        android:fromDegrees="0"
        android:toDegrees="+360"
        android:pivotX="50%"
        android:pivotY="50%"
```

```
36          android:duration = "10000"
37      />
38 </set>
```

① 第 2～38 行设置了一个<set>标签，在该标签内定义了<alpha>、<scale>、<translate>和<rotate>4 个变换标签。

② 第 4～8 行声明了一个设置透明度变换的标签，“0.0”表示不透明，“1.0”表示完全透明。

③ 第 10～20 行声明了一个尺寸变换标签，其中，“0.0”表示大小为 0，“1.0”表示大小为原始尺寸。

④ 第 22～28 行声明了一个位置变换标签。

⑤ 第 30～37 行声明了一个旋转变换标签。旋转从 0°顺时针旋转 360°。

⑥ 注意，每一种变换持续的时间设置，一般透明度变换时间最短，位置和旋转变换持续时间最长。

(4) 设计布局。编写 res/layout 目录下的 main.xml 文件，其代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3       android:orientation = "vertical"
4       android:layout_width = "fill_parent"
5       android:layout_height = "fill_parent"
6       >                                          <! -- 声明一个垂直分布的线性布局 -->
7  < ImageView
8       android:id = "@ + id/iv"
9       android:src = "@drawable/android01"
10      android:layout_width = "wrap_content"
11      android:layout_height = "wrap_content"
12      android:layout_gravity = "center_horizontal"
13      />                                         <! -- 声明一个 ImageView 控件 -->
14     < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
15         android:orientation = "horizontal"
16         android:layout_width = "fill_parent"
17         android:layout_height = "fill_parent"
18         android:gravity = "center_horizontal"
19         >                                       <! -- 声明一个垂直分布的线性布局 -->
20         < Button
21            android:id = "@ + id/btn1"
22            android:text = "图片 1"
23            android:layout_width = "100px"
24            android:layout_height = "wrap_content"
25            android:layout_marginRight = "10dip"
26            />                                   <! -- 声明一个 Button 控件 -->
27     < Button
28            android:id = "@ + id/btn2"
29            android:text = "图片 2"
30            android:layout_width = "100px"
31            android:layout_height = "wrap_content"
32            />                                   <! -- 声明一个 Button 控件 -->
33     </LinearLayout>
34 </LinearLayout>
```

第 9 行设置 ImageView 控件的图片源。注意，使用 Tween Animation 设计动画，XML 文

件只是设置动画的变换属性，不指定图片来源，因此，在布局文件中指定图片源必须是一个图片资源文件。

(5) 开发逻辑代码。打开并编辑 src/cn.com.sgmsc.Tani 包下的文件 TanimationActivity.java，代码如下所示。

```
1  package cn.com.sgmsc.TAni;
2
3  import android.app.Activity;                          //引入相关类
4  import android.os.Bundle;                             //引入相关类
5  import android.view.View;                             //引入相关类
6  import android.view.View.OnClickListener;             //引入相关类
7  import android.view.animation.Animation;              //引入相关类
8  import android.view.animation.AnimationUtils;         //引入相关类
9  import android.widget.Button;                         //引入相关类
10 import android.widget.ImageView;                      //引入相关类
11
12 public class TAnimationActivity extends Activity {
13   @Override
14   public void onCreate(Bundle savedInstanceState) {    //重写 onCreate()方法
15     super.onCreate(savedInstanceState);
16     setContentView(R.layout.main);                     //设置屏幕
17     Button btn1 = (Button)findViewById(R.id.btn1);     //获取 Button 对象
18     btn1.setOnClickListener(new OnClickListener() {
                                          //为 Button 对象添加 OnClickListener 监听器
19       @Override
20       public void onClick(View v) {                     //重写 onClick()方法
21         ImageView iv = (ImageView)findViewById(R.id.iv);
22         iv.setImageDrawable(getResources().getDrawable(R.drawable.android01));
23          Animation animation = AnimationUtils.loadAnimation(TAnimationActivity.this, R.
anim.tween_ani);
24         iv.startAnimation(animation);                   //启动动画
25       }
26     });
27     Button btn2 = (Button)findViewById(R.id.btn2);     //获取 Button 对象
28     btn2.setOnClickListener(new OnClickListener() {
                                          //为 Button 对象添加 OnClickListener 监听器
29       @Override
30       public void onClick(View v) {                     //重写 onClick()方法
31         ImageView iv = (ImageView)findViewById(R.id.iv);
32         iv.setImageDrawable(getResources().getDrawable(R.drawable.android02));
33         Animation animation = AnimationUtils.loadAnimation(TAnimationActivity.this,
           R.anim.tween_ani);
34         iv.startAnimation(animation);                   //启动动画
35       }
36     });
37   }
38 }
```

创建一个最多支持三个流同时播放的，且类型标记为音乐的 SoundPool 对象。第 23、33 行，设置动画的变换属性来源于 tween_ani.xml 文件中所定义的动画属性。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 TweenAnimation 项目。单击"图片 1"按钮后，在屏幕上播放一个由小到大、逐渐清晰的顺时针旋转的图片，如图 8-7 所示。单击"图片 2"按钮后，在屏幕上播放相同的动画，只是这时的图片不同，如图 8-8 所示。







图 8-7 播放补间动画的效果图







图 8-8 播放另一幅图的补间动画

8.3 音频与视频播放

在应用中，适当地运用音频与视频的播放可以收到极好的应用效果。Android 系统提供了对常见格式媒体的编码、解码机制，可以非常容易地集成音频、视频和图片等多媒体文件到应用程序中，例如相册、播放器、录音和摄像等应用程序。当然，有些应用需要硬件的支持。

音频和视频的播放常用到 MediaPlayer 类。该类位于 android.media 包下，在该类中提供了播放、暂停、停止和重复播放等方法。下面分别对音频、视频的播放进行介绍。

8.3.1 播放音频

Android 平台中播放音频有两种方式：使用 SoundPool 类进行播放和使用 MediaPlayer 类进行播放，它们都位于 android.media 包下。前一种方式适合短促且对反应速度比较高的情况(例如播放游戏音效或按键声等)，而后一种方式适合比较长且对时间要求不高的情况(例如播放后台音乐、歌曲等)。Android 的音频文件存放在项目的 res/raw 目录下，其中的 raw 目录需要开发者自己创建。

Android 支持的音频格式有：OGG、MP3、MID、WAV、AMR 等，其中 OGG 格式性能最佳。音频格式采样率分别为 11kHz、22kHz、44.1kHz，16 位立体声。

1. 使用 SoundPool 类播放音频

使用 SoundPool 类播放音频的步骤如下。

1) 创建一个 SoundPool 对象

创建一个 SoundPool 对象的方法为

```
SoundPool(int maxStream, int streamType, int srcQuality)
```

其中，参数 maxStream，指定同时播放的流的最大数量；参数 streamType，指定流的类型，一般

为 STREAM_MUSIC(具体的流类型在 AudioManager 类中列出);参数 srcQuality,指定采样率转化质量,默认值为 0,表示不指定采样率转化质量。

例如,创建一个最多支持三个流同时播放的,且类型标记为音乐的 SoundPool 对象,其代码如下。

```
SoundPool soundPool = new SoundPool(3, AudioManager.STREAM_MUSIC, 0);
```

2) 加载音频资源

SoundPool 可以通过 load()方法来加载一个音频资源,load()方法有 4 种加载方式,分别如下。

(1) 通过一个 AssetFileDescriptor 对象加载音频: int load(AssetFileDescriptor afd, int priority)。

(2) 通过一个资源 ID 加载音频: int load(Context context, int resId, int priority)。

(3) 通过指定的路径加载音频: int load(String path, int priority)。

(4) 通过 FileDescriptor 加载音频: int load(FileDescriptor fd, long offset, long length, int priority)。

一个 SoundPool 能同时管理多个音频,所以可以通过多次调用 load 函数来加载,如果加载成功将返回一个非 0 的 soundID,用于播放时指定特定的音频。一般把多个声音放到 HashMap 中去,例如创建一个最多支持三个流同时播放的,且类型标记为音乐的 SoundPool 对象,并将资源中的 dingdong.ogg 文件以优先级为 1 传入 HasMap 对象中,其代码段如下。

```
SoundPool soundPool = new SoundPool(3, AudioManager.STREAM_MUSIC, 0);
soundPoolMap = new HashMap();
soundPoolMap.put(1, soundPool.load(this, R.raw.dingdong, 1));  //将资源中的音频文件设为优先级 1
```

3) 播放控制

SoundPool 用于控制音频播放的常用方法如表 8-8 所示。

表 8-8 SoundPool 用于控制音频播放的常用方法及说明

方　法	描　述
play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)	播放指定音频的音效,并返回一个 streamID。其中,参数 priority 是流的优先级,值越大优先级越高,影响当同时播放数量超出了最大支持数时 SoundPool 对该流的处理; 参数 loop 是循环播放的次数,−1 为无限循环,其他值为播放 loop+1 次; 参数 rate 是播放的速率,范围为 0.5~2.0(0.5 为一半速率,1.0 为正常速率,2.0 为两倍速率)
pause(int streamID)	暂停指定播放流的音效,参数 streamID 是 play()的返回值,以毫秒为单位
resume(int streamID)	继续播放指定播放流的音效,参数 streamID 是 play()的返回值
stop(int streamID)	终止指定播放流的音效,参数 streamID 是 play()的返回值
setLoop(int streamID, int loop)	设置指定播放流的循环
setVolume (int streamID, float leftVolume, float rightVolume)	设置指定播放流的音量
setPriority(int streamID, int priority)	设置指定播放流的优先级
setRate(int streamID, float rate)	设置指定播放流的速率
unload(int soundID)	卸载一个指定的音频资源,卸载成功返回 true
release()	释放 SoundPool 中的所有音频资源

这里，play()方法传递的是一个 load()返回的 soundID，它指向一个被记载的音频资源。同一个 soundID 可以通过多次调用 play()而获得多个不同的 streamID，当然不要超出同时播放的最大数目。pause()、resume()和 stop()方法是针对播放流操作的，其流的 ID 来自于 play()的返回值。

在 API 中指出，即使调用相关的方法时，使用了无效的 soundID /streamID 也不会导致错误中断。在程序退出时，需要终止播放并释放资源。

2. 使用 MediaPlayer 类播放音频

MediaPlayer 是播放媒体文件最为广泛使用的类。MediaPlayer 可以用来播放大容量的音频文件，同时也可支持播放操作(停止、开始、暂停等)和查找操作的流媒体，它还可支持与媒体操作相关的监听器。

1) MediaPlayer 的状态

使用 MediaPlayer 来播放音频/视频文件或流的控制，是通过一个状态机来管理实现的。一个 MediaPlayer 对象刚被创建或被重新设置时，处于 idle 状态，处于 idle 状态的 MediaPlayer 还没有设置数据源；在向对象中加载了音频/视频资源后便处于 initialized 状态；当对象调用了 prepare()方法后，对象便处于 prepare 状态。对于处于 prepare 状态的对象就可以调用 start()方法来播放音频/视频了。

2) 创建 MediaPlayer 对象

创建 MediaPlayer 对象可以使用两种方式：一种是使用 new MediaPlayer()，另一种是使用 MediaPlayer. create(...)。这两种方法创建的 MediaPlayer 对象所处的状态是不同的，下面分别说明。

使用 new MediaPlayer() 创建对象后，MediaPlayer 对象处于 idle 状态，需要调用 setDataSource() 方法为对象加载音频资源，setDataSource() 支持从 path、URI 和 FileDescriptor 三种途径获取音频/视频资源；然后还需要调用 prepare()方法才能进入等待播放状态。

使用 create()方法创建 MediaPlayer 对象后，MediaPlayer 对象随即处于 prepared 状态，无须调用 prepare()方法，否则系统会报错。这里，create()方法支持从 int(resID)和 URI 两种途径获取音频/视频资源。

MediaPlayer 对象对播放控制的方法基本上与 SoundPool 类中的相同，如 start()、pause()、stop()、seekTo()、setLooping()等方法。需要注意的是，在循环播放的设置上与 SoundPool 的不同，不能指定确定的循环次数，而是一个布尔值，指定是否循环播放。

3) MediaPlayer 监听器

对 MediaPlayer 对象可以定义如下监听器：OnCompletionListener、OnPrepareListener、OnErrorListener、OnBufferingUpdateListener、OnInfoListener、OnVideoSizeChangedListener 和 OnSeekCompleteListener 等。

如果要在播放过程中到达媒体源末端时做某些处理，如从列表中播放下一首歌曲或释放媒体播放器对象等，可以在 OnCompletionListener 监听器里的 onCompletion(MediaPlayer mp)事件中进行编码。如果在准备播放媒体源时需要做某些处理，可在 OnPrepareListener 监听器的 onPrepared(MediaPlayer mp)事件中进行编码。如果在异步操作过程中出现错误时做某些处理(其他错误将在调用方法时抛出异常)，可在 OnErrorListener boolean onError

(MediaPlayer mp, int what, int extra)事件中进行编码,这里,参数 what 指明了已发生错误的类型,它可能为 MEDIA_ERROR_UNKNOWN 或 MEDIA_ERROR_SERVER_DIED。参数 extra 指明了与错误相关的附加信息。

3. 播放音频案例

【案例 8.7】 使用 mediaplayer 和 soundpool 分别播放 mid 和 ogg 音频文件,并可实现两种音频可以同时播放。

【说明】 所有音频文件放在项目的 res/raw 目录下,这个 raw 必须开发者自己创建,但目录名必须是 raw。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 AudioPlay 的 Android 项目。其应用程序名为 AudioPlay,包名为 cn.com.sgmsc.AuPlay,Activity 组件名为 AudioPlayActivity。

(2) 准备音频资源。在项目的 res 中新建目录 raw,并将 backsound.mid 和 dingdong.ogg 音频文件复制到本项目的 res/raw 目录中。

(3) 设计布局。编写 res/layout 目录下的 main.xml 文件,代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
    <TextView
        android:id = "@ + id/textView"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "没有播放任何声音"
    />
    <Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "使用 MediaPlayer 播放声音"
    />
    <Button
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "暂停 MediaPlayer 声音"
    />
    <Button
        android:id = "@ + id/button3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "使用 SoundPool 播放声音"
    />
    <Button
        android:id = "@ + id/button4"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
```

```
            android:text = "暂停 SoundPool 声音"
        />
</LinearLayout>
```

(4) 开发逻辑代码。打开 src/cn. com. sgmsc. AuPlay 包下的 AudioPlayActivity. java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.AuPlay;

import java.util.HashMap;
import android.app.Activity;
import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.SoundPool;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class AudioPlayActivity extends Activity implements OnClickListener{
    Button button1;                                   //4 个按钮的引用
    Button button2;
    Button button3;
    Button button4;
    TextView textView;                                //TextView 的引用
    MediaPlayer mMediaPlayer;
    SoundPool soundPool;                              //SoundPool 声音
    HashMap<Integer, Integer> soundPoolMap;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {    //重写 onCreate()回调方法
        super.onCreate(savedInstanceState);
        initSounds();                                 //初始化声音
        setContentView(R.layout.main);                //设置显示的用户界面
        textView = (TextView) this.findViewById(R.id.textView);    //得到 TextView 的引用
        button1 = (Button) this.findViewById(R.id.button1);          //得到 button 的引用
        button2 = (Button) this.findViewById(R.id.button2);
        button3 = (Button) this.findViewById(R.id.button3);
        button4 = (Button) this.findViewById(R.id.button4);
        button1.setOnClickListener(this);             //为 4 个按钮添加监听
        button2.setOnClickListener(this);
        button3.setOnClickListener(this);
        button4.setOnClickListener(this);
    }

    public void initSounds(){                         //初始化声音的方法
        /* 初始化 MediaPlayer 对象 */
        mMediaPlayer = MediaPlayer.create(this, R.raw.thesamesong);
        /* 初始化 SoundPool 对象 */
        soundPool = new SoundPool(4, AudioManager.STREAM_MUSIC, 100);
        soundPoolMap = new HashMap<Integer, Integer>();
        soundPoolMap.put(1, soundPool.load(this, R.raw.dingdong, 1));
    }
```

```
49      public void onClick(View v) {                  //实现接口中的方法
50          if(v == button1){                          //单击了"使用 MediaPlayer 播放声音"按钮
51              textView.setText("使用 MediaPlayer 播放声音");
52              if(!mMediaPlayer.isPlaying()){
53              mMediaPlayer.start();                  //播放声音
54              }
55          }
56          else if(v == button2){                     //单击了"暂停 MediaPlayer 声音"按钮
57              textView.setText("暂停了 MediaPlayer 播放的声音");
58              if(mMediaPlayer.isPlaying()){
59              mMediaPlayer.pause();                  //暂停声音
60              }
61          }
62          else if(v == button3){                     //单击了"使用 SoundPool 播放声音"按钮
63              textView.setText("使用 SoundPool 播放声音");
64              this.playSound(1, 0);
65          }
66          else if(v == button4){                     //单击了"暂停 SoundPool 声音"按钮
67              textView.setText("暂停了 SoundPool 播放的声音");
68              soundPool.pause(1);                    //暂停 SoundPool 的声音
69          }
70      }
71      /* 用 SoundPoll 播放声音的方法 */
72      public void playSound(int sound, int loop) {
73          AudioManager mgr = (AudioManager)this.getSystemService(Context.AUDIO_SERVICE);
74          float streamVolumeCurrent = mgr.getStreamVolume(AudioManager.STREAM_MUSIC);
75          float streamVolumeMax = mgr.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
76          float volume = streamVolumeCurrent/streamVolumeMax;
77          /* 播放声音 */
78          soundPool.play(soundPoolMap.get(sound), volume, volume, 1, loop, 1f);
79      }
80  }
```

① 第 15 行，定义该类时使用了一个外部 OnClickListener 监听器接口。这么做的目的是，可以在一个 onClick()方法中同时定义多个按钮的响应动作。第 49～70 行是定义 4 个按钮的响应操作代码。

② 第 41～48 行，分别用 MediaPlay 和 SoundPool 两种方式初始化声音方法。由于这里的 SoundPool 对象最多支持 4 个流同时播放，所以需要创建一个 HashMap 来存放多个音频文件。

③ 第 72～79 行，定义了使用 SoundPoll 播放声音的方法。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 AudioPlay 项目，其运行界面如图 8-9 所示。按照按钮上显示的含义，单击按钮即会执行相应的音频控制操作。



图 8-9 AudioPlay 项目的运行界面

8.3.2 播放视频

Android 中播放视频可以通过两种方式来实现。一种是通过 VideoView 组件，该种方式实现起来比较简单容易，但是其可控性不强，可以完成简单的播放任务；另一种是通过 MediaPlayer 在 SurfaceView 进行播放，该种方式实现起来比较麻烦，但是可控性极强。所以

在实际的项目中可以根据不同的需求进行使用。

Android 支持的视频格式有：mp4（MPEG-4 低比特率）、3gp、avi、flv、h. 263、h. 264（avc）等，h. 264 比 h. 263 在相同体积下画质更好。一般地，比较大的文件都存放在 SD 卡中。

1. 使用 VideoView 播放视频

VideoView 类位于 android. widget 包下，它是 Android 自带的、专业化的显示视频的视图控件，它可压缩创建并初始化 MediaPlayer。VideoView 类可从各种源（如资源或内容提供者）加载视频，并且可负责从该视频计算其尺寸，以便其可在任何布局管理器中使用。同样，该类还可提供各种显示选项，如缩放比例和着色，可用来显示 SDCard FileSystem 中存在的视频文件或联机存在的文件。

使用 VideoView 播放视频，首先，需要在 XML 文件中加入 VideoView 控件，然后从 SD Card 中载入 mp4 文件或 3gp 文件，就可以播放视频了。

例如，在把文件 beargolf. 3gp 载入到 SD 卡中后，使用 VideoView 显示视频的简单播放控制应用的代码片段如下。

XML 代码：

```
……
<VideoView
android:id="@+id/vv"
android:layout_width="320px"
android:layout_height="240px"/>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
    <Button
        android:id="@+id/vv_playbtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="播放"/>
    <Button
        android:id="@+id/vv_stopbtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="暂停"/>
</LinearLayout>
……
```

VvPlay. java 代码：

```
……
vv=(VideoView) findViewById(R.id.vv);                       //得到 VideoView 对象 vv
vv_playbtn=(Button)findViewById(R.id.vv_playbtn);
vv_stopbtn=(Button)findViewById(R.id.vv_stopbtn);
vv.setVideoURI(Uri.parse("sdcard/beargolf.3gp"));           //从 SD 卡中加载视频文件
vv.setMediaController(new MediaController(VvPlay.this));  //创建媒体控制器
vv.requestFocus();                                          //VideoView 对象 vv 获得焦点
vv_playbtn.setOnClickListener(new OnClickListener(){
    public void onClick(View v)
    {
```

```
            vv.start();                                    //开始播放视频
        }
    });
    vv_stopbtn.setOnClickListener(new OnClickListener(){
        public void onClick(View v)
        {
            vv.pause();                                    //暂停播放视频
        }
    });
```

从代码中可以看到，使用 VideoView 播放视频比较简单。MediaController 还提供了播放、暂停、滑块等功能。虽然 VideoView 可以很容易地播放视频，但播放位置和播放大小并不受控制，因此，可使用 SurfaceView 来更好地控制播放视频。

2. 使用 MediaPlayer 和 SurfaceView 播放视频

MediaPlayer 主要用于播放音频，它没有提供输出图像的输出界面，这时必须用到 SurfaceView 控件，将它与 MediaPlayer 结合起来，就能达到视频的输出。SurfaceView 类位于 android.view 包下。SurfaceView 常用的方法如表 8-9 所示。

表 8-9　SurfaceView 常用的方法及说明

方　　法	描　　述
getHolder()	得到一个 SurfaceHolder 对象来管理 SurfaceView。返回值是一个 SurfaceHolder 对象
setVisibility(int visibility)	设置是否可见，visibility 取值可以是 VISIBLE、INVISIBLE、GONE，这些常量的含义是： • View.VISIBLE，常量值为 0，意思是可见的。 • View.INVISIBLE，常量值是 4，意思是不可见的。 • View.GONE，常量值是 8，意思是不可见的，并且不占用布局空间

SurfaceHolder 是一个接口，用于管理 SurfaceView，SurfaceHolder 类也位于 android.view 包下，它有两个常用的内部接口 SurfaceHolder.Callback 和 SurfaceHolder.Callback2，Callback2 是实现于 Callback 的。SurfaceHolder.Callback 接口类用于接收预览界面变化的信息，可通过重写以下三个抽象方法来实现。

当 SurfaceView 创建时触发：

```
public abstract void surfaceCreated(SurfaceHolder holder);
```

当 SurfaceView 改变（如预览界面的格式和大小发生改变）时触发：

```
public abstract void surfaceChanged(SurfaceHolder holder, int format, int width, int height);
```

当 SurfaceView 销毁时触发：

```
public abstract void surfaceDestroyed(SurfaceHolder holder);
```

使用 MediaPlayer 和 SurfaceView 的实现步骤如下。

(1) 创建 MediaPlayer 对象，并设置加载的视频文件（使用 setDataSource()方法）。

(2) 在界面布局文件中定义 SurfaceView 控件。

(3) 通过 MediaPlayer.setDisplay(SurfaceHolder sh)来指定视频画面输出到 SurfaceView

之上。

（4）通过 MediaPlayer 的其他一些方法用于播放视频。

下面通过一个视频播放案例来说明其用法。

【案例 8.8】 使用 MediaPlayer 和 SurfaceView 播放一个存放在 SD 上的视频文件，并可以控制视频的播放和暂停。

【说明】 在第 7 章中曾介绍过在模拟器中使用 SD 卡的用法，在这里简单回顾一下。使用 SD 卡中的文件有以下 4 个步骤。

（1）创建一个 SD 卡镜像文件 myvideo.img。在命令行提示状态下输入命令：

```
mksdcard 1024M E:\3gms\myvideo.img
```

（2）关联 SD 卡和模拟器。在 Eclipse 中，启动模拟器，然后选择菜单 Window→Preferences 进入 Preferences 对话框，在 type filter text 中选择 Android→Launch，在右边的 Default emulator options：编辑框中输入“-sdcard e:\4gms\myvideo.img”，设置模拟器默认的启动参数，如图 8-10 所示。

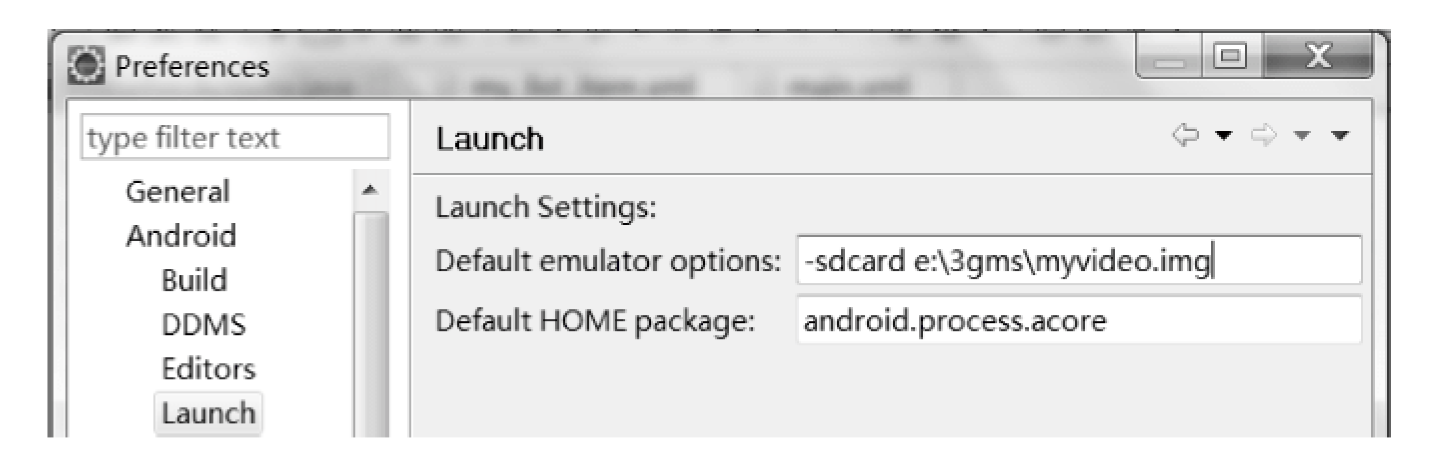


图 8-10　设置模拟器默认的启动参数

关联 SD 卡和模拟器。

（3）向 SD 卡中导入文件。在 Eclipse 的 DDMS 的 FileExplorer 中，选择 sdcard，单击 push 按钮导入，如图 8-11 所示。然后从指定的文件夹中打开需要的视频文件（本例的项目所在文件夹是 e:\3gms\Ch08\VideoPlay，视频文件名为 beargolf.3gp）。

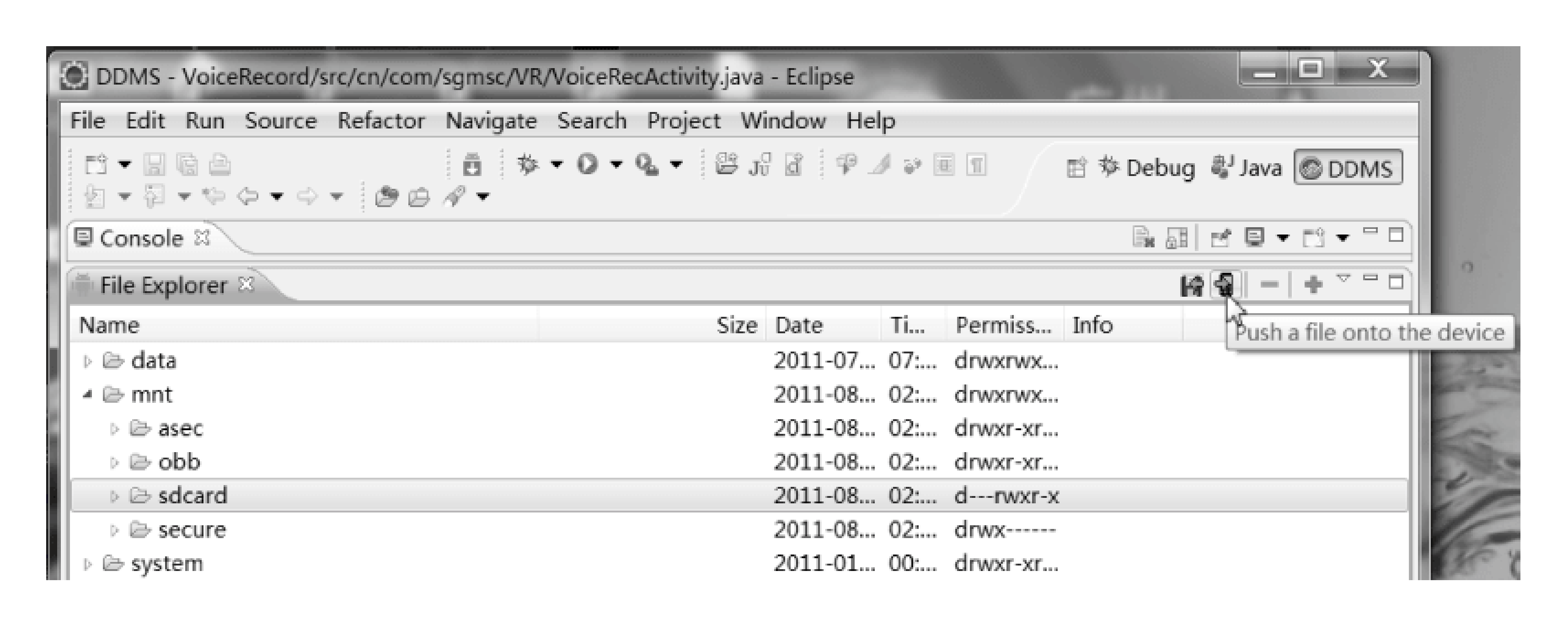


图 8-11　Eclipse 中 DDMS 的 File Explorer

（4）在模拟器中使用 SD 卡中的文件。使用 setDataSource("/sdcard/beargolf.3gp")方法将模拟器的 SD 卡中的文件载入 MediaPlayer 对象中。

【开发步骤及解析】

（1）创建项目。在 Eclipse 中创建一个名为 VideoPlay 的 Android 项目。其应用程序名为 VideoPlay，包名为 cn.com.sgmsc.VPlay，Activity 组件名为 VideoPlayActivity。

(2) 设计布局。编写 res/layout 目录下的 main.xml 文件,在其中需要设置一个 SurfaceView 控件,代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >                                    <!-- 添加一个垂直的线性布局 -->
      <SurfaceView
      android:id="@+id/surfaceView"
      android:layout_width="320px"
      android:layout_height="240px"
        />                                <!-- 添加一个 SurfaceView 用于播放视频 -->
     <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
         >                                <!-- 添加一个线性布局 -->
         <Button
          android:id="@+id/play2_Button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="播放"
                />                        <!-- 添加一个按钮 -->
      <Button
          android:id="@+id/pause2_Button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="暂停"
                />                        <!-- 添加一个按钮 -->
      </LinearLayout>
</LinearLayout>
```

(3) 开发逻辑代码。打开 src/cn.com.sgmsc.Vplay 包下的 VideoPlayActivity.java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.VPlay;

import android.app.Activity;
import android.graphics.PixelFormat;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class VideoPlayActivity extends Activity implements OnClickListener,SurfaceHolder.Callback{
String path = "/sdcard/beargolf.3gp";
Button play_Button;
Button pause_Button;
boolean isPause = false;
SurfaceView surfaceView;
```

```
20 SurfaceHolder surfaceHolder;
21 MediaPlayer mediaPlayer;
22 public void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.main);
25         play_Button = (Button) findViewById(R.id.play2_Button);
26         play_Button.setOnClickListener(this);
27         pause_Button = (Button) findViewById(R.id.pause2_Button);
28         pause_Button.setOnClickListener(this);
29         getWindow().setFormat(PixelFormat.UNKNOWN);
30         surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
31         surfaceHolder = surfaceView.getHolder();
32         surfaceHolder.addCallback(this);
33         surfaceHolder.setFixedSize(176,144);
34         surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
35         mediaPlayer = new MediaPlayer();
36 }
37 public void onClick(View v) {
38     if(v == play_Button){                        //单击"播放"按钮
39         isPause = false;
40         playVideo(path);
41     }
42     else if(v == pause_Button){                  //单击"暂停"按钮
43         if(isPause == false){                    //如果正在播放则将其暂停
44             mediaPlayer.pause();
45             isPause = true;
46         }
47         else{                                    //如果暂停中将继续播放
48             mediaPlayer.start();
49             isPause = false;
50         }
51     }
52 }
53 private void playVideo(String strPath){          //自定义播放影片函数
54     if(mediaPlayer.isPlaying() == true){
55         mediaPlayer.reset();
56     }
57     mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
58     mediaPlayer.setDisplay(surfaceHolder);       //设置 Video 影片以 SurfaceHolder 播放
59     try{
60         mediaPlayer.setDataSource(strPath);
61         mediaPlayer.prepare();
62     }
63     catch (Exception e){
64         e.printStackTrace();
65     }
66     mediaPlayer.start();
67 }
68 public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {}
69 public void surfaceCreated(SurfaceHolder arg0) {}
70 public void surfaceDestroyed(SurfaceHolder arg0) {}
71 }
```

① 第 14 行，定义 VideoPlayActivity 类实现两个接口：OnClickListener 和 SurfaceHolder.Callback。

② 第 15 行，将 SD 卡中文件的路径赋予字符串变量 path 中。

③ 第 29 行，getWindow().setFormat(PixelFormat.UNKNOWN)方法是允许 Android 对窗口显示格式进行其他设置。在编写视频播放程序中这是非常关键的处理。

④ 第 32 行，surfaceHolder.addCallback(this)，为 surfaceHolder 对象添加回调方法。

⑤ 第 33 行，通过 surface 的 holder 设置显示尺寸。

⑥ 第 34 行，设置显示类型，参数 SURFACE_TYPE_PUSH_BUFFERS 是一系统定义的 int 型常量，表示 SurfaceView 本身不包含数据源，其数据来源于其他对象。

⑦ 第 53～67 行定义了 playVideo()方法。其中，第 54～56 行是设置当视频正在播放时将 mediaPlayer 对象初始化；第 57 行是设置音频为流类型；第 58 行是设置画面的输出对象；第 61 行设置播放视频文件的数据源；第 62 行设置 mediaPlayer 对象处于 prepare 状态；第 66 行设置开始播放视频。

⑧ 第 68～70 行是重写 SurfaceHolder.Callback 接口的三个抽象方法，在这里并没有定义具体代码。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 VideoPlay 项目，单击“播放”按钮即可以看到视频，同时可听到视频的配音，单击“暂停”按钮可暂停视频的播放；如果正在播放时再单击“播放”按钮，视频将从头开始播放。其运行界面如图 8-12 所示。

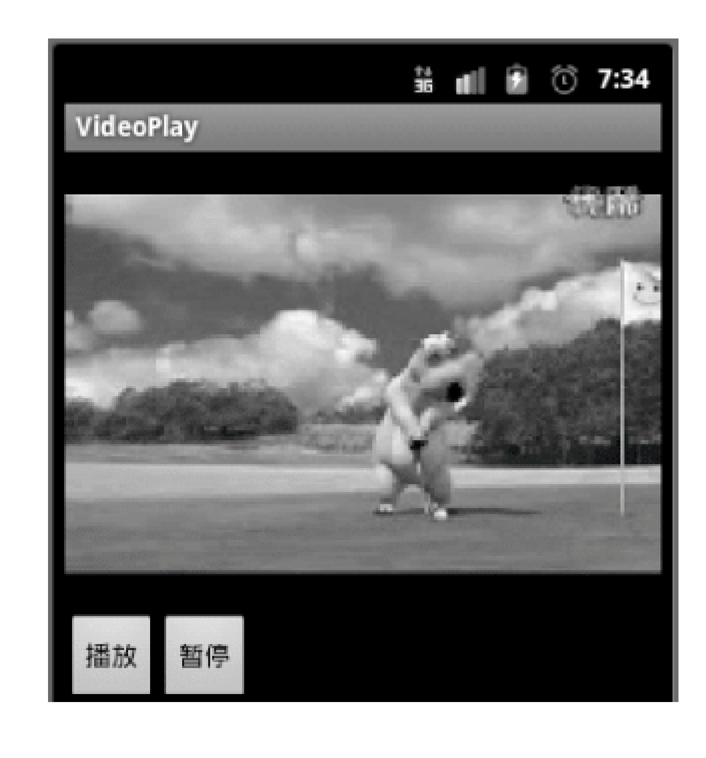


图 8-12 能控制播放或暂停视频播放器

8.4 声音与图像数据采集

8.4.1 声音采集

Android 的声音采集(即录音)可使用 MediaRecorder 类来实现。MediaRecorder 类位于 android.media 包下，它包含 Audio 和 Video 的记录功能。

使用 MediaRecorder 录制的语音文件格式通常为.amr，录制的文件一般都是保存在 SD 卡中。使用 MediaRecorder 进行录音通常需要编写下列代码段。

```
MediaRecorder recorder = new MediaRecorder();            //创建一个 MediaRecorder 对象 recorder
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);            //设置麦克风
recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);    //设置输出格式
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);       //设置音频编码 Encoder?
recorder.setOutputFile("/sdcard/myrec/myrecord01.amr");            //设置音频文件保存路径
recorder.start();                                        //开始录制
```

注意，录音操作需要拥有录音权限和对存储卡的写入权限。所以一定要记住，在编写录音的应用程序时，要在 AndroidManifest.xml 文件的“<manifest>”标签内加入下列权限代码。

```
<uses-permission android:name = "android.permission.RECORD_AUDIO"/>
<uses-permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
```

下面通过一个实用案例来说明录音程序的设计。

【案例 8.9】 使用 VoiceRecord 类实现一个录音器，该录音器可以进行录音、停止、播放、删除功能，且将录制的文件存放在 SD 卡中。

【说明】 使用 SD 卡保存录制的文件，首先在移动设备（手机或模拟器）上要有 SD 卡。判断设备中是否有 SD 卡，使用下列方法：

```
Environment.getExternalStorageState().equals(android.os.Environment.MEDIA_MOUNTED)
```

如果返回的值为 true，则设备上存在 SD 卡，否则没有 SD 卡。

【开发步骤及解析】

(1) 用户界面设计。在屏幕的上端并排放置 4 个按钮，分别执行录音、停止、播放、删除的监听操作。在屏幕的下部用一个列表列出已录制的声音文件，在按钮和列表之间是一行文本用于显示当前的声音文件。使用背景色来划分分区。

(2) 创建项目。在 Eclipse 中创建一个名为 VoiceRecord 的 Android 项目。其应用程序名为 VoiceRecord，包名为 cn.com.sgmsc.VR，Activity 组件名为 VoiceRecActivity。

(3) 准备图片。将用于图片按钮的图片资源复制到本项目的 res/drawable-mdpi 目录中。

(4) 创建颜色资源。创建并编写 res/values 目录下的颜色描述文件 color.xml，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <color name = "black">#FF000000</color>
    <color name = "lightpurple">#FFbdb2e6</color>
    <color name = "white">#FFFFFFFF</color>
</resources>
```

(5) 设计布局。编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@color/lightpurple">
    <LinearLayout
        android:id = "@ + id/LinearLayout01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
        <ImageButton
            android:id = "@ + id/ImageButton01"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/record">
        </ImageButton>
        <ImageButton
            android:id = "@ + id/ImageButton02"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/stop">
        </ImageButton>
        <ImageButton
```

```
            android:id = "@ + id/ImageButton03"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/play">
        </ImageButton>
        <ImageButton
            android:id = "@ + id/ImageButton04"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/delete">
        </ImageButton>
    </LinearLayout>
    <TextView
        android:id = "@ + id/TextView01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:textColor = "@color/black">
    </TextView>
    <ListView
        android:id = "@ + id/ListView01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:background = "@color/black">
    </ListView>
</LinearLayout>
```

(6) 设计 ListView 的条目布局。编写 res/layout 目录下的 my_list_item. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<CheckedTextView
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:id = "@ + id/myCheckedTextView1"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent"
  android:textColor = "@color/white"/>
```

这个 xml 文件中只定义了一个 CheckedTextView 控件。CheckedTextView 类继承 TextView 并实现 Checkable 接口。

(7) 开发逻辑代码：打开 src/cn. com. sgmsc. VR 包下的 VoiceRecActivity. java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.VR;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import android.app.Activity;
import android.content.Intent;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.AdapterView;
```

```java
import android.widget.ArrayAdapter;
import android.widget.CheckedTextView;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class VoiceRecActivity extends Activity{
  private ImageButton myButton1;
  private ImageButton myButton2;
  private ImageButton myButton3;
  private ImageButton myButton4;
  private ListView myListView1;
  private String strTempFile = "voice_";
  private File myRecAudioFile;
  private File myRecAudioDir;
  private File myPlayFile;
  private MediaRecorder mMediaRecorder;
  private ArrayList<String> recordFiles;
  private ArrayAdapter<String> adapter;
  private TextView myTextView1;
  private boolean sdCardExit;
  private boolean isStopRecord;

  @Override
  public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    myButton1 = (ImageButton) findViewById(R.id.ImageButton01);
    myButton2 = (ImageButton) findViewById(R.id.ImageButton02);
    myButton3 = (ImageButton) findViewById(R.id.ImageButton03);
    myButton4 = (ImageButton) findViewById(R.id.ImageButton04);
    myListView1 = (ListView) findViewById(R.id.ListView01);
    myTextView1 = (TextView) findViewById(R.id.TextView01);
    myButton2.setEnabled(false);
    myButton3.setEnabled(false);
    myButton4.setEnabled(false);

    /* 判断 SD Card 是否插入 */
    sdCardExit = Environment.getExternalStorageState().equals(
        android.os.Environment.MEDIA_MOUNTED);
    /* 取得 SD Card 路径作为录音的文件位置 */
    if (sdCardExit)
        myRecAudioDir = Environment.getExternalStorageDirectory();

    /* 取得 SD Card 目录里的所有.amr 文件 */
    getRecordFiles();

    adapter = new ArrayAdapter<String>(this,
        R.layout.my_list_item, recordFiles);
    /* 将 ArrayAdapter 添加到 ListView 对象中 */
    myListView1.setAdapter(adapter);

    /* 录音 */
    myButton1.setOnClickListener(new ImageButton.OnClickListener(){
```

```
            //......省略一：省略了该方法的代码段，相应代码将在后面给出解析
        });

        /* 停止 */
        myButton2.setOnClickListener(new ImageButton.OnClickListener(){
          //......省略二：省略了该方法的代码段，相应代码将在后面给出解析
        });

        /* 播放 */
        myButton3.setOnClickListener(new ImageButton.OnClickListener(){
          //......省略三：省略了该方法的代码段，相应代码将在后面给出解析
        });

        /* 删除 */
        myButton4.setOnClickListener(new ImageButton.OnClickListener(){
          //......省略四：省略了该方法的代码段，相应代码将在后面给出解析
        });

        myListView1.setOnItemClickListener(
          new AdapterView.OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1,
                int arg2, long arg3){
              /* 当单击列表中的文件名时将"删除"及"播放"按钮置为 Enable */
              myButton3.setEnabled(true);
              myButton4.setEnabled(true);
              myPlayFile = new File(myRecAudioDir.getAbsolutePath()
                            + File.separator
                            + ((CheckedTextView) arg1).getText());
              myTextView1.setText("你选的是："
                            + ((CheckedTextView) arg1).getText());
           }
      });
    }

    @Override
    protected void onStop(){
    if (mMediaRecorder != null && !isStopRecord){
        /* 停止录音 */
        mMediaRecorder.stop();
        mMediaRecorder.release();
        mMediaRecorder = null;
    }
    super.onStop();
   }

   /* 向 ArrayList 对象中添加 SD 卡中的.amr 文件名 */
   private void getRecordFiles(){
     //......省略五：省略了该方法的代码段，相应代码将在后面给出解析
   }

   /* 打开播放录音文件的程序 */
   private void openFile(File f) {
      //......省略六：省略了该方法的代码段，相应代码将在后面给出解析
   }
```

```
125
126 }
```

① 第 54、58 行判断外部存储设备 SD 卡是否插入设备中。其中 android. os. Environment. MEDIA_MOUNTED 是系统环境常量，表示 SD 卡已插入。方法 Environment. getExternalStorageDirectory()的功能是取 SD 卡上保存文件的所在路径。这里，用变量 myRecAudioDir 保存这个路径，以便后面用于保存录音文件时使用。

② 第 63～66 行实例化一个 ArrayAdapter 对象 adapter 并设置，其数据源由自定义的 getRecordFiles()取出，并赋值到一个 ArrayList 对象 recordFiles 中。getRecordFiles()方法的代码稍后解析。

③ 第 88～103 行定义 ListView 的条目被单击的监听器。ListView 条目列出的是录音文件的文件名，当单击 ListView 中的文件名时，“播放”及“删除”按钮置为 Enable。new File (myRecAudioDir. getAbsolutePath()是获取保存文件的路径，File. separator 是获得文件路径的字符，((CheckedTextView) arg1). getText()是获得文件的文件名。

④ 第 106～114 行定义了 onStop()方法。在该方法内停止录音，并释放 MediaRecorder 对象。接下来对省略的部分代码作解析。

省略一：这部分省略的代码是定义“录音”按钮的单击监听。具体代码如下。

```
/* 录音 */
myButton1.setOnClickListener(new ImageButton.OnClickListener(){
  @Override
  public void onClick(View arg0){
    try{
      if (!sdCardExit){
          Toast.makeText(VoiceRecActivity.this, "请插入 SD Card",
             Toast.LENGTH_LONG).show();
          return;
       }

       /* 创建录音频文件 */
       myRecAudioFile = File.createTempFile(strTempFile, ".amr", myRecAudioDir);
       mMediaRecorder = new MediaRecorder();
       /* 设置录音来源为麦克风 */
       mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
       mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
       mMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
       mMediaRecorder.setOutputFile(myRecAudioFile.getAbsolutePath());
       mMediaRecorder.prepare();
       mMediaRecorder.start();
       myTextView1.setText("录音中");
       myButton2.setEnabled(true);
       myButton3.setEnabled(false);
       myButton4.setEnabled(false);
       isStopRecord = false;
     }
     catch (IOException e){
       e.printStackTrace();
     }
   }
});
```

① 第 6～10 行是判断是否插入了 SD 卡，如果没有插入 SD 卡就给出提示并返回，退出录音操作。

② 第 13 行通过 createTempFile()得到一个录制文件的文件名。这个文件名以 voice_***.amr 命名，文件路径为变量 myRecAudioDir 指定的值。该方法的格式为：

```
createTempFile(String prefix, String suffix, File directory)
```

其功能是在指定目录 directory 中创建一个新的空文件，其中，参数 directory 是指定的目录路径，文件名由给定的前缀 prefix 和后缀 suffix 字符串生成，前缀和后缀之间由任意产生的一组数据构成，以便区分由该方法产生的其他文件名。

③ 第 16～21 行设置录音的来源、格式、音频编码、音频文件保存路径，设置完成后开始录制操作。

④ 第 23～25 行设置按钮的可用状态。

省略二、省略三、省略四：这部分省略的代码是定义按钮“停止”、“播放”和“删除”的监听器。具体代码如下。

```
/* 停止 */
myButton2.setOnClickListener(new ImageButton.OnClickListener(){
  @Override
  public void onClick(View arg0){
    if (myRecAudioFile != null){
        /* 停止录音 */
        mMediaRecorder.stop();
        mMediaRecorder.release();
        mMediaRecorder = null;
        /* 将音频文件名给 Adapter */
        adapter.add(myRecAudioFile.getName());
        myTextView1.setText("停止: " + myRecAudioFile.getName());
        myButton2.setEnabled(false);
        isStopRecord = true;
      }
    }
});

/* 播放 */
myButton3.setOnClickListener(new ImageButton.OnClickListener(){
  @Override
  public void onClick(View arg0){
    if (myPlayFile != null && myPlayFile.exists()){
        /* 打开播放的程序 */
        openFile(myPlayFile);
      }
  }
});

/* 删除 */
myButton4.setOnClickListener(new ImageButton.OnClickListener(){
  @Override
  public void onClick(View arg0){
    if (myPlayFile != null){
        /* 先将 Adapter 删除文件名 */
        adapter.remove(myPlayFile.getName());
        /* 删除文件 */
```

```
38          if (myPlayFile.exists())
39            myPlayFile.delete();
40          myTextView1.setText("完成删除");
41          /* 判断 adapter 为空时按钮的状态 */
42          if (adapter.isEmpty()){
43            myButton3.setEnabled(false);
44            myButton4.setEnabled(false);
45          };
46        }
47      }
48    });
49
```

① 第 7~9 行，如果正在录音，则停止录音并释放 MediaRecorder 对象。第 11、12 行是从适配器中取出当前的音频文件名，并在 TextView 对象中显示停止信息。

② 第 23~25 行设置，如果文件存在，则播放音频文件。方法 openFile()是打开指定的文件，即播放指定文件。

③ 第 36 行是删除适配器中指定的数据条目；第 39 行删除文件；第 42~45 行是根据 adapter 中的状态来确定按钮的可用状态。

省略五的代码定义了 getRecordFiles()方法，省略六的代码定义了播放音频的播放对话框。具体代码如下。

```
1  /* 向 ArrayList 对象中添加 SD 卡中的.amr 文件名 */
2  private void getRecordFiles(){
3    recordFiles = new ArrayList<String>();
4    if (sdCardExit){
5        File files[] = myRecAudioDir.listFiles();
6        if (files != null){
7           for (int i = 0; i < files.length; i++){
8             if (files[i].getName().indexOf(".") >= 0){
9               /* 只取.amr 文件 */
10              String fileS = files[i].getName().substring(files[i].getName().indexOf("."));
11              if (fileS.toLowerCase().equals(".amr"))
12                recordFiles.add(files[i].getName());
13            }
14          }
15        }
16      }
17 }
18
19 /* 打开播放录音文件的程序 */
20 private void openFile(File f) {
21   Intent intent = new Intent();
22   intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
23   intent.setAction(android.content.Intent.ACTION_VIEW);
24   intent.setDataAndType(Uri.fromFile(f),"audio/*");
25   startActivity(intent);
26 }
27
```

① 第 5 行得到一个文件型的数组。myRecAudioDir.listFiles()是获得指定路径中的文件列表。

② 第 10 行，files[i].getName().substring(files[i].getName().indexOf("."))获得文件

的扩展名。

③ 第 11、12 行，将扩展名为“. amr”的文件加入到 recordFiles 中，recordFiles 是一个 ArrayList 对象。

④ 第 20～26 行，打开一个音频播放的进度条对话框。

(8) 添加权限。打开根目录下的 AndroidManifest. xml，添加权限，其代码如下所示。

```
1 <?xml version = "1.0" encoding = "utf - 8"?>
2 < manifest xmlns:android = "http://schemas.android.com/apk/res/android"
3         package = "cn.com.sgmsc.VR"
4         android:versionCode = "1"
5         android:versionName = "1.0.0">
6      < application
7          android:icon = "@drawable/icon"
8          android:label = "@string/app_name">
9          < activity android:name = ".VoiceRecActivity"
10                   android:label = "@string/app_name">
11              < intent - filter >
12                  < action
13                  android:name = "android.intent.action.MAIN" />
14                  < category
15                  android:name = "android.intent.category.LAUNCHER" />
16              </intent - filter >
17          </activity >
18      </application >
19      < uses - permission android:name = "android.permission.RECORD_AUDIO"/>
20      < uses - permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
21 </manifest >
```

① 第 19 行添加录音权限。

② 第 20 行添加存储卡的可写入权限。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 VoiceRecord 项目。初始运行项目时，只有“录音”按钮可用。单击“录音”，开始录制声音；单击“停止”，即停止录音，并按指定的文件名前缀、后缀形成文件名，以列表的方式显示出文件名，如图 8-13 所示。选择该文件名列表中的一文件名，即刻在列表的上方出现选中项的信息，并且“播放”或“删除”按钮呈可用状态。此时，可执行“播放”或“删除”操作，例如，单击“播放”按钮，则开始播放声音，如图 8-14 所示。



图 8-13 已录制了 4 个音频文件



图 8-14 正在播放第二个音频文件

8.4.2 图像采集

Android 的图像采集可以用两种实现方式，一是利用手机自带的 Camera 应用程序；二是使用 Camera 类获得摄像头信息。一般情况下，采集的图像文件都会保存在存储卡上，因此，需要在 AndroidManifest. xml 中添加存储卡可写权限，即在＜manifest＞标签内增加权限代码：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

请注意，Android 的模拟器不支持 Camera 拍照的图像显示，而只是会在显示区出现网格，以表示模拟器的 Camera 被初始化了，拍照后获得的图像会用一张系统内部的图片代替显示。而在真机上，对摄像头的应用就能正常显示，不会出现模拟器中的这种问题。因此，开发的这类应用，必须在真机上进行测试。

1. 利用手机自带的 Camera

每个 Android 手机都有一个自带的 Camera 应用程序，可以实现一般的拍照功能。程序员可以在自己的应用中直接调用系统自带的 Camera 应用程序来完成图像采集功能。下面通过一个案例来说明如何使用手机自带的 Camera 开发图像采集应用。

【案例 8. 10】 通过一个简单界面调用模拟器内置的 Camera 应用程序实现拍照应用。

【说明】 通过用户界面的一个按钮来调用 Android 自带的 Camera 应用程序，与拍照相关的调节操作由 Camera 应用程序完成。拍照完毕后，向用户界面返回照片。

当用户界面调用系统应用时，该用户的 Activity 将会转入后台，而当 Camera 应用完毕后，用户的 Activity 将重新被激活。因为用户的 Activity 在其后台期间，有可能会因系统内存吃紧被销毁，所以在程序中需要使用生命周期方法 onSaveInstanceState()和 onRestoreInstanceState()来保存状态。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 AndCamera 的 Android 项目。其应用程序名为 And_Camera，包名为 cn. com. sgmsc. andcamera，Activity 组件名为 AndCameraActivity。

(2) 设计布局。编写 res/layout 目录下的 main. xml 文件，这是一个用户自己定义的界面，用于启动 Android 的自带 Camera 程序和显示使用自带 Camera 程序拍摄的照片；当自带 Camera 程序退出时，该界面又处于活动状态。代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_horizontal|center_vertical">

    <RelativeLayout android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="6"
        android:gravity="center_horizontal|center_vertical">

        <TextView android:id="@+id/field"
```

```
14              android:layout_width = "fill_parent"
15              android:layout_height = "wrap_content"
16              android:text = "No Image"
17              android:gravity = "center_horizontal"/>
18
19          < ImageView android:id = "@ + id/image"
20              android:layout_width = "fill_parent"
21              android:layout_height = "wrap_content"/>
22
23      </RelativeLayout >
24
25      < Button android:id = "@ + id/button"
26          android:layout_width = "fill_parent"
27          android:layout_height = "wrap_content"
28          android:text = "Take Photo"
29          android:layout_weight = "1"/>
30  </LinearLayout >
```

① 第 6 行设置水平居中和垂直居中。如果为一个属性设置多个值,可以用"|"作分隔符。

② 在<RelativeLayout>布局中定义了两个控件 TextView 和 ImageView,在代码中没有设置这两个控件的相对位置,那么它们都位于 RelativeLayout 布局的中央。在 Java 代码中要控制二者的可见或隐藏状态。

(3) 开发逻辑代码。打开 src/cn. com. sgmsc. andcamera 包下的 AndCameraActivity. java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.andcamera;

import java.io.File;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

public class AndCameraActivity extends Activity {
protected Button u_button;
protected ImageView u_image;
protected TextView u_field;
protected String u_path;                          //拍照后,保存文件的文件全名
protected boolean u_taken;                        //拍照操作的状态
protected static final String PHOTO_TAKEN   = "photo_taken";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 设定屏幕显示为横向 */
        this.setRequestedOrientation(0);    //设置纵向显示为: setRequestedOrientation(90)
```

```
        setContentView(R.layout.main);

        u_image = ( ImageView ) findViewById( R.id.image );
        u_field = ( TextView ) findViewById( R.id.field );
        u_button = ( Button ) findViewById( R.id.button );
        u_button.setOnClickListener( new ButtonClickHandler() );
        /*图片文件路径*/
        u_path = Environment.getExternalStorageDirectory() + "/take_picture.jpg";
    }

    public class ButtonClickHandler implements View.OnClickListener {
      public void onClick( View view ){
          File file = new File( u_path );
          Uri outputFileUri = Uri.fromFile( file );
          /*使用 Android 自带的 Camera 捕捉图像程序*/
          Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE );
          intent.putExtra( MediaStore.EXTRA_OUTPUT, outputFileUri );
          startActivityForResult( intent, 0 );
      }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
      switch( resultCode ) {
            case 0:
                break;
            case -1:
                onPhotoTaken();                  //显示捕获的照片
                break;
      }
    }
    /*显示捕获的照片*/
    protected void onPhotoTaken() {
      u_taken = true;
      BitmapFactory.Options options = new BitmapFactory.Options();
      options.inSampleSize = 2;                  //设置图片的容量大小为原始大小的1/2
      Bitmap bitmap = BitmapFactory.decodeFile( u_path, options );
      u_image.setImageBitmap(bitmap);
      u_field.setVisibility( View.GONE );
    }

    /*生命周期方法：当系统要销毁 activity 之前调用*/
    @Override
    protected void onSaveInstanceState( Bundle outState ) {
      outState.putBoolean( AndCameraActivity.PHOTO_TAKEN, u_taken );
    }
    /*生命周期方法：如果 Activity 在后台没有因为运行内存吃紧被清理，
      * 则切换回时会触发 onRestoreInstanceState 方法
      */
    @Override
    protected void onRestoreInstanceState( Bundle savedInstanceState){
      if( savedInstanceState.getBoolean( AndCameraActivity.PHOTO_TAKEN ) ) {
            onPhotoTaken();
      }
   }
```

```
85
86 }
```

① 第 29 行设置屏幕的方向为横向的。因模拟器一般是纵向显示的，而拍照取景一般是横向的，所以在 setContentView(R. layout. main)；执行之前要设定屏幕的显示方向。设置屏幕为横向还可以使用下列方式：

```
this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE)
```

② 第 37 行设置拍照的图片文件保存的路径及文件名。这里指定的是保存在 SD 卡中。

③ 第 45 行是创建一个 Intent，这个 Intent 是 Android 自带的 Camera 捕捉图像应用程序。这里，android. provider. MediaStore. ACTION_IMAGE_CAPTURE 表示是捕捉图像，如果参数是 android. provider. MediaStore. ACTION_VIDEO_CAPTURE，则表示是捕捉视频。第 47 行是启动 Camera 程序。

④ 第 52～60 行重写了 onActivityResult()。当子模块事情做完之后返回主界面，需要接着处理由子模块返回的数据，可在回调方法 onActivityResult()中定义实现。该方法的参数 requestCode 用于判断是从哪个 Activity 返回的，如果值≥0，通常表示从用户定义的子模块中返回，如果值＝－1，表示从系统的子模块中返回。在本例中，当 requestCode＝－1 时表示从 Camera 应用程序中返回，那么在返回后需要调用 onPhotoTaken()方法来显示由 Camera 捕获的图片。

⑤ 第 62～69 行定义 onPhotoTaken()方法，显示由 Camera 捕获的照片。第 65 行是确定图片显示的大小。当 options. inSampleSize ＝ 1 时，设置图片的容量大小为原始大小；当 options. inSampleSize ＝ 2 时，设置图片的容量大小为原始大小的 1/2。从 main. xml 中知道，图片控件 u_image 与文本控件 u_field 是在同一位置上布局的，因此，在显示图片时，文本 u_field 必须设置为不可见，且不占布局空间，所以在第 68 行调用方法：

```
u_field.setVisibility( View.GONE );
```

⑥ 第 73～75 行重写生命周期方法 onSaveInstanceState()，此方法在系统要销毁 Activity 之前调用，用于保存该 Activity 的状态。

⑦ 第 80～84 行重写生命周期方法 RestoreInstanceState()，此方法在 Activity 被销毁后重新启用时调用，用于恢复该 Activity 的状态。

(4) 添加权限。打开根目录下的 AndroidManifest. xml，添加权限，其代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "cn.com.sgmsc.andcamera"
    android:versionCode = "1"
    android:versionName = "1.0" >

    <uses - sdk android:minSdkVersion = "10" />

    <application
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name" >
        <activity
            android:label = "@string/app_name"
            android:name = ".AndCameraActivity" >
```

```
15            <intent-filter>
16                <action android:name="android.intent.action.MAIN" />
17
18                <category android:name="android.intent.category.LAUNCHER" />
19            </intent-filter>
20        </activity>
21    </application>
22
23 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
24
25 </manifest>
```

第 23 行添加存储卡的可写入权限。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 AndCamera 项目。初始运行时可以看到一个横置的界面，对于模拟器，还需要按下 Ctrl＋F12 键将模拟器横置（第 1 章中已介绍），如图 8-15 所示。单击 Take Photo 按钮即可调用系统的 Camera 程序，如图 8-16 所示。在按下 Camera 的快门键之后进行拍照，并且可以即时浏览拍照的效果，如图 8-17 所示。如果不满意可以单击 RETAKE 按钮重新拍照或单击 CANCEL 按钮删除照片，如果满意则单击 OK 按钮，返回主界面，并可以在界面上显示所拍的照片，如图 8-18 所示。注意，在真机上运行效果更好。

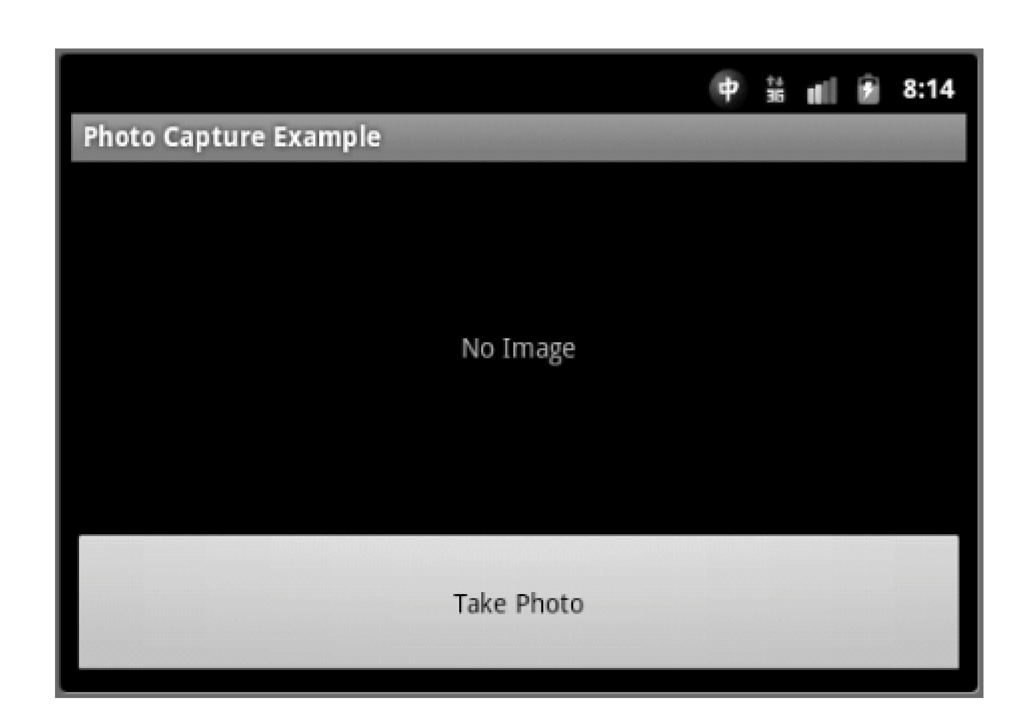


图 8-15　初始运行时的界面

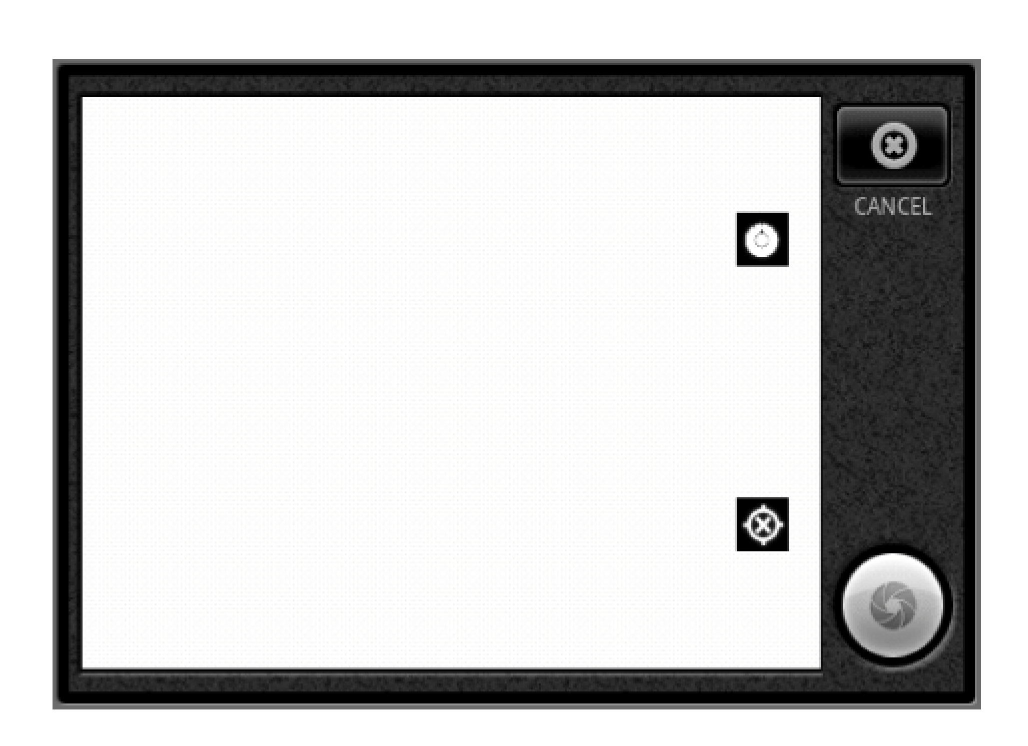


图 8-16　调用 Camera 程序界面

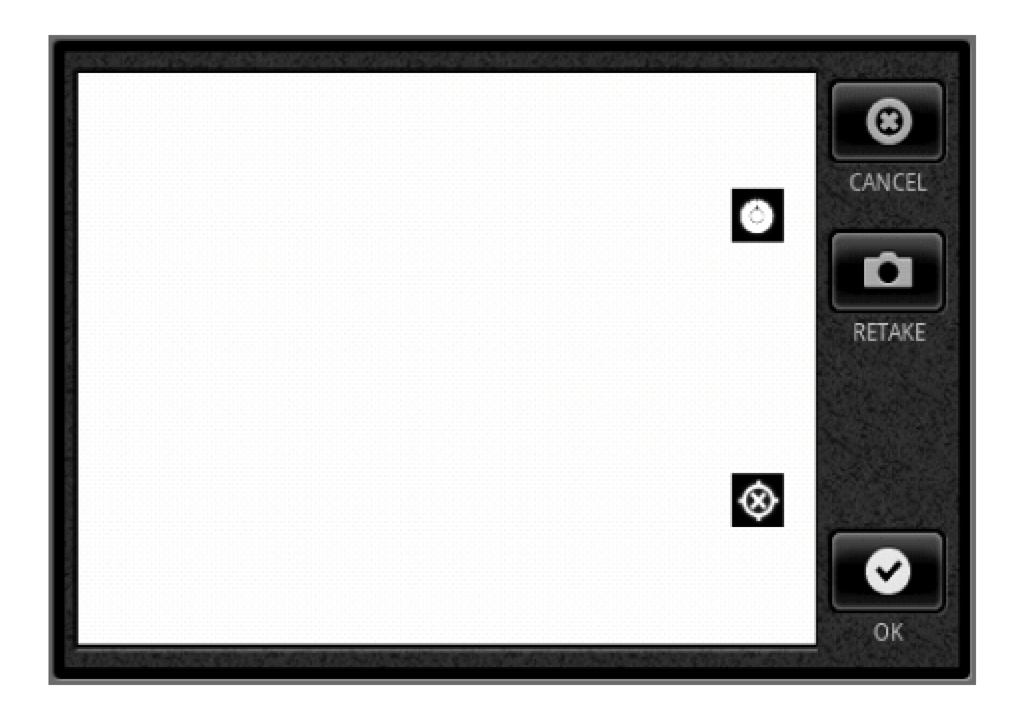


图 8-17　拍照后的 Camera 界面

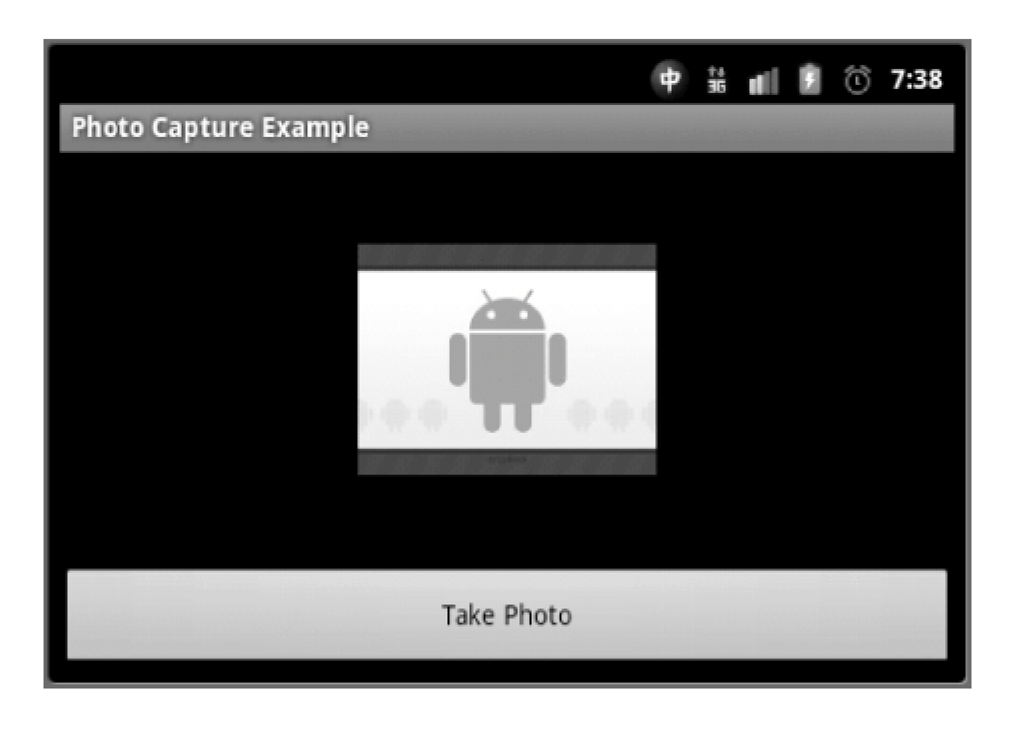


图 8-18　返回主界面显示照片

本例是使用 Camera 进行拍照，也可以使用 Camera 进行摄像应用。但是由系统自带的 Camera 应用程序的界面是不能修改的，且按钮是英文显示的。如果想按照用户自己的意图来

设计拍照界面，就需要用到 Camera 类由程序员编程设计。

2. 使用 Camera 类

Android 的 Camera 类位于 android.hardware 包下，Camera 类包含取景器(viewfinder)和拍摄照片的功能。使用 Camera 类开发图像采集应用时，需要在布局文件中添加 SurfaceView 视图，作为摄像头屏幕，在代码程序中要实现 SurfaceHolder.Callback 接口。Camera 类常用的方法如表 8-10 所示。

表 8-10 Camera 类常用的方法及说明

方　法	描　述
open()	使用静态 open()方法访问 Camera 服务
release()	应用程序完成服务后，使用 release()释放连接
setParameters ()	照相机参数设置
startPreview ()	打开照相机预览功能
stopPreview ()	停止照相机预览
takePicture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback jpeg)	实现拍照。其中，第一个参数为快门，回调方法 onShutter()在快门关闭后立即触发；第二个参数为 raw 数据，图片原始数据通过 byte[]传入回调方法；第三个参数为 jpeg 格式的数据，图片数据通过 byte[]传入回调方法

下面通过案例来说明如何使用 Camera 类来开发自己的图像采集应用。

【案例 8.11】 使用 Camera 类设计一个简单的拍照应用，要求可以实现拍照、重拍、保存和预览功能，拍照文件以 take_picture ***.jpg 命名，保存在 SD 卡的 CameraTest 下。

【说明】 使用 Camera 类需要在布局中添加一个 SurfaceView 视图。与案例 8.8 一样，用 Camera 控制取景拍照，用 SurfaceView 显示预览。

用 Java 开发一个专门用于图片处理的类，负责图片的裁剪、缩放、合并、保存等功能。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 PicCamera 的 Android 项目。其应用程序名为 PictureCamera，包名为 cn.com.sgmsc.PCamera，Activity 组件名为 PicCameraActivity。

(2) 设计布局。编写 res/layout 目录下的 main.xml 文件，将按钮设置在屏幕的中部，以此作分隔，上半部作为取景视窗，下半部作为浏览视窗。代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
     android:orientation = "vertical"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent"
     android:gravity = "center_horizontal"
     ><!-- 添加一个垂直的线性布局 -->
     <SurfaceView
       android:id = "@ + id/mySurfaceView"
       android:gravity = "center_horizontal"
       android:layout_width = "wrap_content"
       android:layout_height = "200px"
     /><!-- 添加一个 SurfaceView 用于浏览 -->
     <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
```

```
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:gravity = "center_horizontal"
            ><!-- 添加一个线性布局 -->
            <ImageButton
              android:id = "@ + id/camerabtn"
              android:textSize = "18px"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:src = "@drawable/camera1"
            /><!-- 添加"打开"按钮 -->
            <ImageButton
              android:id = "@ + id/savedbtn"
              android:textSize = "18px"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:src = "@drawable/save1"
            /><!-- 添加"保存"按钮 -->
            <ImageButton
              android:id = "@ + id/redobtn"
              android:textSize = "18px"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:src = "@drawable/redo1"
            /><!-- 添加"拍照"按钮 -->
      </LinearLayout>
      <ImageView
        android:id = "@ + id/myImageView"
        android:layout_gravity = "center"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
</LinearLayout>
```

(3) 开发逻辑代码。打开 src/cn. com. sgmsc. Pcamera 包下的 PicCameraActivity. java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.PCamera;

import java.io.IOException;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.PixelFormat;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;
import android.hardware.Camera.ShutterCallback;
import android.os.Bundle;
import android.os.Environment;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.View.OnTouchListener;
```

```
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;
import android.widget.ImageView;

public class PicCameraActivity extends Activity implements SurfaceHolder.Callback {

    /** 当 Activity 首次创建时调用 */
    private Bitmap[] bmps = new Bitmap[6];
    private Camera myCamera;
    private ImageButton mButCamera, mButSaved, mButRedo;
    private ImageView myImageView;

    //定义 private 对象
    private SurfaceView mSurfaceView;
    private SurfaceHolder myholder;
    private AutoFocusCallback mAutoFocusCallback = new AutoFocusCallback();
    private String path = "CameraTest";    //定义文件夹名
    private String name = "take_picture";  //定义照片文件名前缀
    private Bitmap bmp = null;
    private int count;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 设置窗口全屏显示 */
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                                  WindowManager.LayoutParams.FLAG_FULLSCREEN);
        /* 隐藏标题栏 */
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);

        mSurfaceView = (SurfaceView) findViewById(R.id.mySurfaceView);
        myholder = mSurfaceView.getHolder();
        myholder.addCallback(PicCameraActivity.this);
        myholder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

        myImageView = (ImageView) findViewById(R.id.myImageView);

        mButCamera = (ImageButton) findViewById(R.id.camerabtn);
        mButSaved = (ImageButton) findViewById(R.id.savedbtn);
                                            //“搜索”按钮(1、保存文件,读取上传)
        mButRedo = (ImageButton) findViewById(R.id.redobtn);
                                            //“重拍”按钮(放弃保存文件,并重新拍照)

        /* 按钮效果处理 */
        mButCamera.setOnTouchListener(new OnTouchListener() {
          //…… 省略一,省略了该方法的代码段,相应代码将在后面给出解析
        });

        mButSaved.setOnTouchListener(new OnTouchListener() {
          //…… 省略二,省略了该方法的代码段,相应代码将在后面给出解析
        });

        mButRedo.setOnTouchListener(new OnTouchListener() {
```

```
            //…… 省略三,省略了该方法的代码段,相应代码将在后面给出解析
           });

          /* 拍照按钮的事件处理 */
          mButCamera.setOnClickListener(new Button.OnClickListener() {
            //…… 省略四,省略了该方法的代码段,相应代码将在后面给出解析
          });

          /* 保存按钮的事件处理 */
          mButSaved.setOnClickListener(new Button.OnClickListener() {
            //…… 省略五,省略了该方法的代码段,相应代码将在后面给出解析
          });

          /* 重拍按钮的事件处理 */
          mButRedo.setOnClickListener(new Button.OnClickListener() {
            //…… 省略六,省略了该方法的代码段,相应代码将在后面给出解析
          });
      }
      //@Override
      public void surfaceCreated(SurfaceHolder surfaceholder) {
          try {
              /* 打开相机 */
              myCamera = myCamera.open();
              myCamera.setPreviewDisplay(myholder);
          } catch (IOException exception) {
              myCamera.release();
              myCamera = null;
      }
    }
    //@Override
    public void surfaceChanged(SurfaceHolder surfaceholder, int format, int w, int h) {
        /* 相机初始化 */
        initCamera();
        count++;
    }
     //@Override
    public void surfaceDestroyed(SurfaceHolder surfaceholder) {
        stopCamera();
        myCamera.release();
        myCamera = null;
    }

    ShutterCallback myShutterCallback = new ShutterCallback(){
        //…… 省略七,省略了该方法的代码段,相应代码将在后面给出解析
    };
    PictureCallback myRawCallback = new PictureCallback(){
        //…… 省略八,省略了该方法的代码段,相应代码将在后面给出解析
    };
    PictureCallback myjpegCallback = new PictureCallback(){
        //…… 省略九,省略了该方法的代码段,相应代码将在后面给出解析
    };

    public final class AutoFocusCallback implements android.hardware.Camera.AutoFocusCallback {
        //…… 省略十,省略了该内部类的代码段,相应代码将在后面给出解析
    }
```

```
128    private void takePicture() {
129        //…… 省略十一,省略了该方法的代码段,相应代码将在后面给出解析
130    };
131    private ShutterCallback shutterCallback = new ShutterCallback() {
132        public void onShutter() {
133            /* 按下快门瞬间会调用这里的程序 */
134        }
135    };
136    private PictureCallback rawCallback = new PictureCallback() {
137        public void onPictureTaken(byte[] _data, Camera _camera) {
138
139        }
140    };
141    private PictureCallback jpegCallback = new PictureCallback() {
142        //…… 省略十二,省略了该方法的代码段,相应代码将在后面给出解析
143    };
144
145    /* 相机初始化的方法 */
146    private void initCamera() {
147        if (myCamera != null) {
148            try {
149                Camera.Parameters parameters = myCamera.getParameters();
150                parameters.setPictureFormat(PixelFormat.JPEG);
151                parameters.setPictureSize(1024, 768);
152                name = "take_picture";
153
154                parameters.setPreviewSize(200, 200);//屏幕大小
155                myCamera.setParameters(parameters);
156                myCamera.setPreviewDisplay(myholder);
157                myCamera.startPreview();
158            } catch (Exception e) {
159                e.printStackTrace();
160            }
161        }
162    }
163
164    /* 停止相机的方法 */
165    private void stopCamera() {
166        if (myCamera != null) {
167            try {
168                /* 停止预览 */
169                myCamera.stopPreview();
170            } catch (Exception e) {
171                e.printStackTrace();
172            }
173        }
174    }
175 }
```

① 第 46、47 行设置窗口全屏显示。在 Android 中设置窗口的显示属性可使用方法 getWindow().setFlags()。如本行语句是设置窗口全屏：

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                     WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

设置窗口始终点亮：

```
EgetWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
                      WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

设置窗口背景模糊：

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_BLUR_BEHIND,
                       WindowManager.LayoutParams.FLAG_BLUR_BEHIND);
```

② 第 52～55 行，在 Activity 的 OnCreate 函数中设置好 SurfaceView，包括设置 SurfaceHolder.Callback 对象和 SurfaceHolder 对象的类型。

③ 第 92～101 行，重写 SurfaceHolder.Callback 的 surfaceCreated()方法。第 95 行使用 Camera 的 open()方法开启摄像头硬件，这个 API 在 SDK 2.3 之前是没有参数的，2.3 以后支持多摄像头，所以开启前可以通过 getNumberOfCameras()方法先获取摄像头数目，再通过 getCameraInfo()方法得到需要开启的摄像头 ID，然后传入 open 函数开启摄像头，假如摄像头开启成功则返回一个 Camera 对象，否则就抛出异常。第 96 行，setPreviewDisplay 为摄像头设置 SurfaceHolder 对象。

④ 第 103～107 行，重写 SurfaceHolder.Callback 的 surfaceChanged()方法。在该方法中初始化 Camera，其具体操作在方法 initCamera()中定义。

⑤ 第 146～162 行定义了 initCamera()方法。在该方法中设置了 Camera 摄取的图片的文件格式、图片大小、文件名，以及预览的屏幕大小等属性。第 157 行 startPreview()开始预览。

⑥ 第 165～174 行定义了 stopCamera()方法。在该方法中停止相机的预览功能。

接下来对省略的部分代码作解析。

省略一、省略二、省略三：这部分省略的代码是定义三个按钮被按下或抬起的效果，当按下按钮时按钮图片呈按下图标，释放按钮时按钮图片呈常态图标。具体代码如下。

```
/* 按钮效果处理 */
mButCamera.setOnTouchListener(new OnTouchListener() {

    //@Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            //更改为按下时的背景图片
            bmps[0] = BitmapFactory.decodeResource(getResources(),
                                                    R.drawable.camera2);
            mButCamera.setImageBitmap(bmps[0]);
        } else if (event.getAction() == MotionEvent.ACTION_UP) {
            //改为抬起时的图片
            bmps[1] = BitmapFactory.decodeResource(getResources(),
                                                    R.drawable.camera1);
            mButCamera.setImageBitmap(bmps[1]);
        }
        return false;
    }
});

mButSaved.setOnTouchListener(new OnTouchListener() {
    //…… 此处代码与 mButCamera.setOnTouchListener()中的代码定义相似，在此省略
});

```

```
25 mButRedo.setOnTouchListener(new OnTouchListener() {
26     //…… 此处代码与 mButCamera.setOnTouchListener()中的代码定义相似,在此省略
27 });
```

① 第 2～19 行是定义拍照按钮 mButCamera 的按钮效果，当按下按钮时按钮图片呈 bmps[0]中的图标，当释放按钮时按钮图片呈 bmps[1]中的图标。

② 其余的按钮效果定义与 mButCamera 的按钮效果相似，当按下 mButSaved 按钮时图片呈 bmps[2]中的图标，当释放该按钮时按钮图片呈 bmps[3]中的图标；当按下 mButRedo 按钮时图片呈 bmps[4]中的图标，当释放该按钮时按钮图片呈 bmps[5]中的图标。具体代码在此省略。

省略四、省略十、省略十一、省略十二：这部分省略的代码是定义拍照事件的处理。具体代码如下。

```
        /* 拍照按钮的事件处理 */
        mButCamera.setOnClickListener(new Button.OnClickListener() {
            //@Override
            public void onClick(View arg0) {
                /* 设置按钮可见性并拍照 */
                  mButCamera.setVisibility(View.GONE);
                  mButRedo.setVisibility(View.GONE);
                  if (bmps[0] != null && !bmps[0].isRecycled())
                      bmps[0].recycle();
                  /* 对焦拍照 */
                  myCamera.autoFocus(mAutoFocusCallback);  //调用 AutoFocusCallback 类
                  myImageView.setImageBitmap(null);
              }
          });

//…… 省略部分代码

        public final class AutoFocusCallback implements android.hardware.Camera.AutoFocusCallback {
            public void onAutoFocus(boolean focused, Camera camera) {
                /* 对到焦点拍照 */
                if (focused) {
                    takePicture();
                } else
                      mButCamera.setVisibility(View.VISIBLE);
            }
        }
        private void takePicture() {
            if (myCamera != null) {
                myCamera.takePicture(shutterCallback, rawCallback, jpegCallback);
            }
        }
        private ShutterCallback shutterCallback = new ShutterCallback() {
            public void onShutter() {
                /* 按下快门瞬间会调用这里的程序 */
            }
        };
        private PictureCallback rawCallback = new PictureCallback() {
            public void onPictureTaken(byte[] _data, Camera _camera) {

            }
```

```
41        };
42        private PictureCallback jpegCallback = new PictureCallback() {
43            public void onPictureTaken(byte[ ] _data, Camera _camera) {
44                /* 取得相片 */
45                try {
46                    /* 设定 Button 可视性 */
47                    mButCamera.setVisibility(View.GONE);
48                    mButSaved.setVisibility(View.VISIBLE);
49                    mButRedo.setVisibility(View.VISIBLE);
50                    /* 取得相片 Bitmap 对象 */
51                    bmp = BitmapFactory.decodeByteArray(_data, 0, _data.length);
52                } catch (Exception e) {
53                    e.printStackTrace();
54                }
55            }
56        };
```

① 第 2～14 行定义按钮 mButCamera 的 OnClickListener 监听器，在其 onClick ()方法中设置按钮的可见性，第 8、9 行设置释放“拍照”按钮，这里的语句作用是：如果 Bitmap 数组元素 bmps[0]没有被系统回收就执行回收，为内存提供空间。第 11 行设置自动对焦拍照。第 12 行设置 ImageView 控件为初始状态。

② 第 11 行将通过对象 mAutoFocusCallback 调用 AutoFocusCallback 类中定义的代码。

③ 第 18～26 行定义 AutoFocusCallback 类。其中第 21、22 行代码是定义对焦后即刻调用 takePicture()方法。

④ 第 27～31 行定义 takePicture()方法。只要 myCamera 对象存在即调用 myCamera 的实现拍照方法 takePicture(shutterCallback，rawCallback，jpegCallback)。在快门关闭后立即触发回调方法 onShutter()，该方法定义在第 32～36 行；图片原始数据通过 byte[]传入回调方法，该方法定义在第 37～41 行；图片为 jpeg 格式的数据，图片数据通过 byte[]传入回调方法，该方法定义在第 42～56 行，其中第 51 行将拍得的照片传入 Bitmap 对象 bmp 中。

省略五、省略六、省略七、省略八、省略九：这部分省略的代码是定义保存和重拍事件的处理。具体代码如下。

```
        /* “保存”按钮的事件处理 */
        mButSaved.setOnClickListener(new Button.OnClickListener() {
            //@Override
            public void onClick(View arg0) {
              mButSaved.setVisibility(View.GONE);
              mButRedo.setVisibility(View.GONE);
              /* 保存文件 */
              if (bmp != null) {
                /* 检查 SD Card 是否存在 */
                if (!Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorage State())) {
                  /* SD 卡不存在，显示 Toast 信息 */
                  Toast.makeText(PicCameraActivity.this, "SD 卡不存在!上传失败!",
                                     Toast.LENGTH_LONG).show();
                } else {
                    try {
                      ImageTool.saveImage(bmp,path,name,100);
                      myCamera.takePicture(myShutterCallback, myRawCallback, myjpegCallback);
                      Toast.makeText(PicCameraActivity.this, "相片保存成功!",
                                   Toast.LENGTH_LONG).show();
```

```
20                  } catch (Exception e) {
21                    e.printStackTrace();
22                  }
23              }
24            }
25            if (!bmp.isRecycled())
26              bmp.recycle();
27            mButCamera.setVisibility(View.VISIBLE);
28            /* 重新设定 Camera */
29            stopCamera();
30            initCamera();
31        }
32      });
33
34      /* "重拍"按钮的事件处理 */
35      mButRedo.setOnClickListener(new Button.OnClickListener() {
36          //@Override
37          public void onClick(View arg0) {
38              mButCamera.setVisibility(View.VISIBLE);
39              mButSaved.setVisibility(View.GONE);
40              mButRedo.setVisibility(View.GONE);
41              /* 重新设定 Camera */
42              if (!bmp.isRecycled())
43                  bmp.recycle();
44              stopCamera();
45              initCamera();
46          }
47      });
48
49 //…… 省略部分代码
50
51 ShutterCallback myShutterCallback = new ShutterCallback(){
52       @Override
53       public void onShutter(){}
54 };
55 PictureCallback myRawCallback = new PictureCallback(){
56       @Override
57       public void onPictureTaken(byte[] data, Camera camera) {}
58 };
59 PictureCallback myjpegCallback = new PictureCallback(){
60       @Override
61       public void onPictureTaken(byte[] data, Camera camera) {
62            Bitmap bm = BitmapFactory.decodeByteArray(data, 0, data.length);
63            myImageView.setImageBitmap(bm);              //将图片显示到下方的 ImageView 中
64       }
65 };
```

① 第 2～32 行定义了"保存"按钮 OnClickListener 监听。在 onClick()方法中设置按钮的可见状态；判断 SD 卡是否存在，如果存在则调用 ImageTool 类的 saveImage()方法将 bmp 中的图片保存到指定的 SD 卡的路径中，并再次调用 takePicture()方法；然后回收 bmp 对象，设置"拍照"按钮可见，重新设定 Camera。

② 第 17 行是再次调用 takePicture()方法的语句。该语句的作用是当拍照完成后即刻调用 myShutterCallback 对象的回调方法，该方法定义在第 51～54 行；得到图片后；图片原始

数据通过 byte[]传入回调方法，该方法定义在第 55～58 行；图片为 jpeg 格式的数据，图片数据通过 byte[]传入回调方法，该方法定义在第 59～65 行，主要执行将拍照所得图片显示在下方的 ImageView 控件中。

③ 第 35～47 行定义了“重拍”按钮 OnClickListener 监听。在 onClick()方法中设置按钮的可见状态，回收 bmp 对象，重新设定 Camera。

(4) 开发 ImageTool 类代码。在 src/cn. com. sgmsc. Pcamera 包下创建 ImageTool. java 文件并编辑之。这个自定义类包括对图片的常用操作处理方法，有一定实用价值。在此只对本例将用到的 saveImage()方法作解析，省略了其余部分代码，完整代码可参见本书下载资源指定网址上的源代码。代码如下所示。

```
package cn.com.sgmsc.PCamera;

import java.io.File;
import java.io.FileOutputStream;
import java.util.Calendar;
import java.util.Date;
import android.graphics.Bitmap;
import android.graphics.Matrix;
import android.os.Environment;
import android.util.Log;

public class ImageTool {
    //切割图片的一部分
    public static Bitmap imageCut(Bitmap bmp, int left, int top, int right, int bottom) {
        //…… 省略了该方法的代码段，参见本书源代码
    }

    //把图片按固定比例缩小
    public static Bitmap imageZoom(Bitmap bmp, int iWidth, int iHeight) {
        //…… 省略了该方法的代码段，参见本书源代码
    }

    //图片合并
    public static Bitmap imageMerge(Bitmap[ ] bmps) {
        //…… 省略了该方法的代码段，参见本书源代码
    }

    //保存图片
    public static void saveImage(Bitmap bmp, String path, String filename, int quality) {
        String time = callTime();
        if (bmp != null) {
            try {
                /* 文件夹不存在就创建 */
                File f = new File(Environment.getExternalStorageDirectory(),path);
                if (!f.exists()) {
                    f.mkdir();
                }
                /* 保存相片文件 */
                File n = null;
                n = new File(f, filename + time + ".jpg");
                FileOutputStream bos = new FileOutputStream(n.getAbsolutePath());
                /* 文件转换 */
```

```
43                  bmp.compress(Bitmap.CompressFormat.JPEG, quality, bos);
44                  /* 调用 flush()方法,更新 BufferStream */
45                  bos.flush();
46                  /* 结束 OutputStream */
47                  bos.close();
48              } catch (Exception e) {
49                  e.printStackTrace();
50              }
51          }
52          if(!bmp.isRecycled())
53              bmp.recycle();
54      }
55
56      //获取系统时间
57      public static String callTime() {
58          //…… 省略了该方法的代码段,参见本书源代码
59      }
60 }
```

① 第 34～37 行定义从指定的路径中获取文件夹，判断该文件夹是否存在，如果不存在则建立。

② 第 39、40 行定义在指定的文件夹中创建一个以 filename + time + ".jpg"命名的文件。其中参数 filename 从 PicCameraActivity 类中得到，其值为"filename"，由此构成的文件名为 take_picture ***.jpg，符合案例的要求。

③ 第 41～47 行定义了从 bmp 中将照片转换入 take_picture ***.jpg 文件中。

④ 第 52、53 行是回收 bmp 对象。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 PicCamera 项目。初始运行时界面如图 8-19 所示。单击"拍照"按钮，照片在按钮上方显示，可见"保存"和"重拍"按钮，如图 8-20 所示。单击"保存"按钮，照片在按钮下方显示，如果保存成功则提示"相片保存成功!"，如图 8-21 所示。注意，在真机上运行效果更好。



图 8-19 初始运行界面

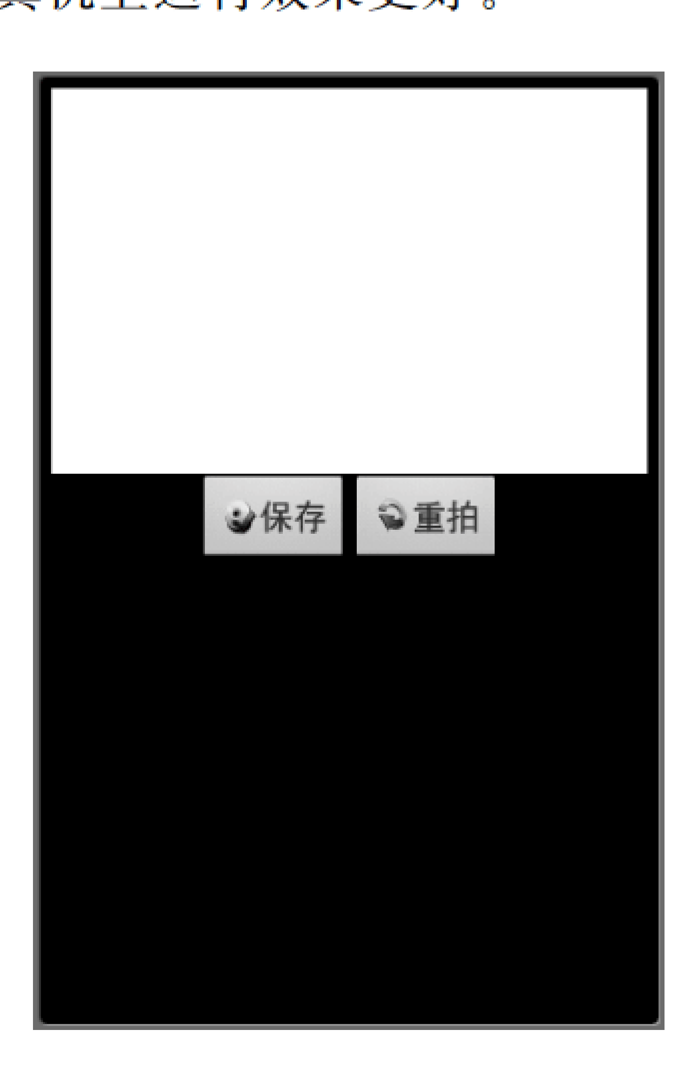


图 8-20 单击"拍照"按钮后



图 8-21 单击"保存"按钮后

小结

本章对运行于 Android 平台上的相关多媒体技术作了集中介绍。主要以案例的形式说明关于二维、三维图形应用，动画应用，音频、视频的播放应用，录制声音和拍摄图像等应用的编程实现方法。由于 Android 的版本升级较快，对多媒体技术的支持度也随着版本的升级而更新，所以，在实践中，读者还需不断学习，不断地丰富对多媒体应用的编程技巧，掌握其更精湛的技术。

在第 8.3 节中，介绍了音频文件的播放应用。在 Android 中对音频的播放和管理一般都与 Service 有关，涉及后台的处理操作控制问题，这是第 9 章中将介绍的内容，请继续学习。



练习

设计微博的拍照功能，并以一定的规律为照片文件命名，保存到 SD 卡中。



第9章 Android后台处理

在 Android 的应用中,有一些应用是没有界面的,并且可以在运行其他应用时同时运行这些不需要界面的应用。例如,播放音乐,接收、发送和广播消息,线程内的数据控制等。这些都是在 Android 后台中运行着的代码,不需要在屏幕显示中作布局设计。

9.1 消息提示

Android 系统提供一套友好的消息提示机制,不会打断用户当前的操作。常用的方式有 Toast 和 Notification。下面分别介绍这两种消息提示的用法。

9.1.1 Toast

Toast 类在 android.widget 包下。它向用户提供一种快速的即时消息,消息内容简短。当 Toast 被显示时,悬浮于应用程序的最上方,没有焦点,停留几秒钟后便会自动消失。Toast 在程序中常用于某项操作执行后是否成功的消息提示。

Toast 对象的创建通过 makeText()方法实现。makeText()方法有两种格式,格式一:makeText(Context context, int resId, int duration),格式二:makeText(Context context, String message, int duration)。其中,第一个参数是 Toast 对象当前的上下文引用;第二个参数在格式一中是一个字符串的资源标识符,在格式二中是一个字符串,其字符串的内容是将要显示的消息内容;第三个参数是指定 Toast 显示的时间长度,通常使用 Toast 的常量 LENGTH_LONG(值为 1,显示时间较长),或者 LENGTH_SHORT(值为 0,显示时间较短)来表示。在完成 Toast 对象创建后通过 show()方法即可将消息提示显示在屏幕上。

Toast 的默认显示方式是在屏幕的下半部位置,且只有文本消息。可以使用 Toast 的一些方法改变其显示效果,例如,将 Toast 显示位置设置到屏幕中部,可使用如下代码。

```
Toast toast = Toast.makeText(getApplicationContext(),"自定义位置 Toast", Toast.LENGTH_LONG);
toast.setGravity(Gravity.CENTER, 0, 0);
toast.show();
```

【案例 9.1】 完善第 7 章案例 7.2 的功能。在系统完成对数据库表操作后给出提示消息,反馈操作完成的情况。

【说明】 本例只对删除日志给出 Toast 消息提示,其余部分留作练习。

【开发步骤及解析】

(1) 复制已有项目。复制项目 ZSWB_Diary2 为 ZSWB_Diary3:在 Eclipse 的 Project

Explorer 中，选择项目 ZSWB_Diary2，然后选择菜单 Edit→Copy，再选择菜单 Edit→Paste，打开 Copy Project 对话框，在 Project name：编辑框内输入“ZSWB_Diary3”，如图 9-1 所示。单击 OK 按钮即可完成项目的复制操作。

（2）修改代码。展开 ZSWB_Diary3 项目，修改 src/cn. com. sgmsc. ZSWB 包下的代码文件 DiaryListActivity. java。增加的代码或与原代码不同处使用粗体显示，代码如下所示。

```
package cn.com.sgmsc.ZSWB;

import android.widget.Toast;
//…… 省略部分引入相关类的代码

public class DiaryListActivity extends Activity {
  //…… 省略部分代码

  //方法：删除指定的日志记录
  public void deleteDiary(int id){                    //id 为要删除记录的 id
    SQLiteDatabase db = myHelper.getWritableDatabase();  //获得数据库对象
    try{
      int count = db.delete(TABLE_NAME, ID + " = ?", new String[]{id + ""});
      db.close();
      if(count == 1){                                 //删除成功
          Toast.makeText(DiaryListActivity.this, "删除日志成功!", Toast.LENGTH_LONG).show();
          getBasicInfo(myHelper);                     //重新获取数据库信息
          myAdapter.notifyDataSetChanged();           //刷新 ListView
      }
      else{                                           //删除失败
          Toast.makeText(DiaryListActivity.this, "删除失败,请重试!", Toast.LENGTH_LONG).show();
      }
    }catch(Exception e){
              e.printStackTrace();
    }
  }
}
```

【运行结果】 运行 KDWB_Diary3 项目。首先选择一条日志，按下手机（或模拟器）中的 menu 键，调出选项菜单，单击“删除”菜单项执行删除操作，当对数据库表的操作完成后，会显示提示消息，如图 9-2 所示。

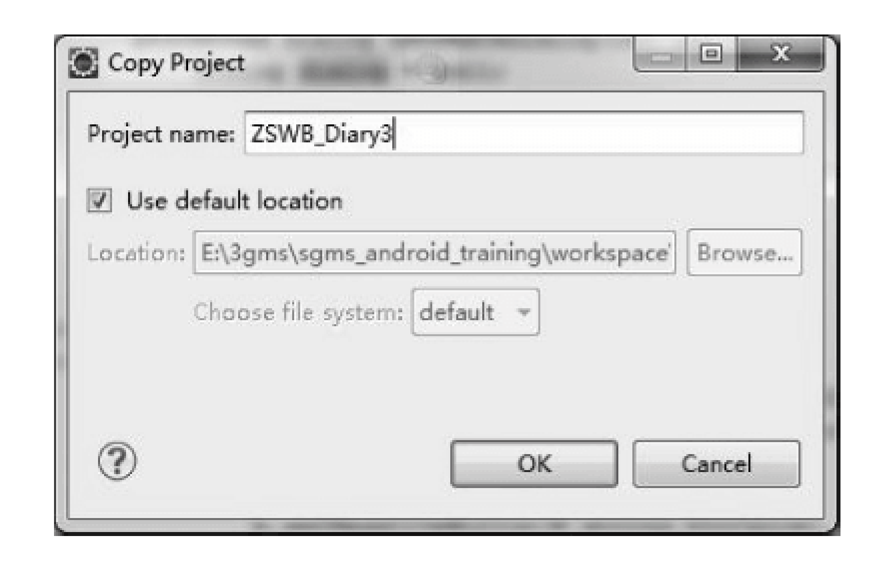




图 9-1 “复制项目”对话框



图 9-2 执行删除操作之后的 Toast 提示消息

9.1.2 Notification

Notification 类在 android. app 包下。它是另一种提示消息的方式，Notification 无须

Activity，将消息内容显示在手机状态条中。手机的状态条位于屏幕的最上方，通常显示电池电量、信息强度等信息。在Android手机中，用手指按住状态条往下拉，即可打开状态条查看系统的提示消息，如图9-3所示。

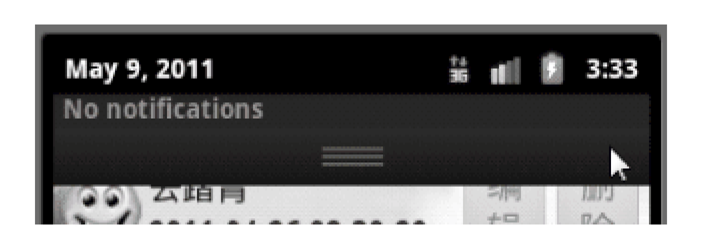


图9-3 打开的状态条显示

在Android中，Notification包含如下功能：创建新的状态栏图标，在扩展的状态条窗口显示额外的信息(也可以发起一个Intent)，闪烁/LED，让手机振动，发出声音(铃声、媒体库歌曲)等。

所有的Notification对象由NotificationManager来管理，NotificationManager类也位于android.app包下。NotificationManager(通知管理器)是用来处理Notification的系统服务，使用getSystemService()方法可以获得对它的引用。通过使用NotificationManager，可以触发新的Notification，修改现有的Notification或者删除那些不再需要的Notification。

1. NotificationManager和Notification中的常用方法和属性

NotificationManager类主要负责将Notification在状态条显示出来和取消，常用的方法如表9-1所示。

表9-1 NotificationManager中常用的方法及说明

方法	描述
cancel(int id)	取消以前显示的一个Notification，如果是一个短暂的Notification，试图将其隐藏，如果是一个持久的Notification，将从状态条中移走
cancelAll()	取消以前显示的一个所有Notification
getSystemService(NOTIFICATION_SERVICE)	初始化一个NotificationManager对象
notify(int id, Notification notification)	把Notification持久地发送到状态条上

Notification类常用的方法如表9-2所示。

表9-2 Notification中常用的方法及说明

方法	描述
Notification()	创建一个默认属性的Notification对象
Notification(int icon, CharSequence tickerText, long when)	创建指定图标、显示文本内容、显示时间的一个Notification对象。一般是立即显示，第三个参数取System.currentTimeMillis()
setLatestEventInfo (Context context, CharSequence contentTitle, CharSequence contentText, PendingIntent contentIntent)	设置显示在拉伸状态栏中的Notification对象的属性，单击后将发送PendingIntent对象。其中，第一个参数为上下文；第二个参数为Notification对象的标题；第三个参数为Notification对象的内容；第四个参数为PendingIntent对象，单击后将发送所包含的intent

问一下：

什么是PendingIntent？它与Intent有什么不同？

PendingIntent类是一个Intent的描述，它位于android.app包下。Pending一词的含义是即将发生或来临的事情。PendingIntent这个类用于处理即将发生的事情。例如在通知

Notification 中用于跳转页面，但不是马上跳转。

Intent 是即时启动，随所在的 Activity 消失而消失。而 PendingIntent 可以看作是对 Intent 的包装，通常通过 getActivity()、getBroadcast()、getService()来得到 PendingIntent 的实例，当前 Activity 并不能马上启动它所包含的 Intent，而是在外部执行 PendingIntent 时，调用 Intent 的。Intent 一般是用作 Activity、Sercvice、BroadcastReceiver 之间传递数据，而 PendingIntent 一般用在 Notification 上，或其他的方法参数中。

Notification 里面有很多属性，设置这个类主要是设置 Notification 的相关属性。下面是 Notification 使用的常用属性。

defaults：状态条的默认属性。

icon：状态条的图标。

sound：来通知时的提示音。

tickerText：显示在状态条中的提示文本。

vibrate：来通知时振动。

when：来通知时的时间。

在设置属性值时，Notification 经常使用系统定义的一些常量，如：DEFAULT_ALL，使用所有属性的默认值；DEFAULT_LIGHTS，使用默认闪烁提示；DEFAULT_SOUND，使用默认提示声音；DEFAULT_VIBRATE，使用手机振动；等等。注意，模拟器不支持振动，在程序测试时应该在手机设备中测试带振动的这部分功能。

2. 使用 Notification 和 NotificationManager 的基本步骤

Notification 和 NotificationManager 操作相对比较简单，一般是先获取一个 NotificationManager 对象，然后实例化 Notification，设置它的属性，最后通过 NotificationManager 对象发出通知就可以了。具体的步骤如下。

(1) 获取 NotificationManager 对象。

```
String ns = Context.NOTIFICATION_SERVICE;
NotificationManager mNotificationManager = (NotificationManager) getSystemService(ns);
```

(2) 创建一个 Notification 对象。

```
Notification notification = new Notification();
notification.icon = R.drawable.notification_icon;
```

或者，使用稍微复杂一些的方法创建 Notification 对象。

```
int icon = R.drawable.notification_icon;          //通知图标
CharSequence tickerText = "Hello";                //状态栏(Status Bar)显示的通知文本提示
long when = System.currentTimeMillis();           //通知产生的时间,会在通知信息里显示
Notification notification = new Notification(icon, tickerText, when);
```

(3) 设置 Notification 的各个属性，如内容、图标、标题、相应的动作处理等。

① 设置在状态条(Status Bar)中显示的通知文本提示，如：

```
notification.tickerText = "hello";
```

② 设置发出提示音，如：

```
notification.defaults |= Notification.DEFAULT_SOUND;
```

```
2  notification.sound = Uri.parse("file:///sdcard/notification/ringer.mp3");
3  notification.sound = Uri.withAppendedPath(Audio.Media.INTERNAL_CONTENT_URI, "6");
```

第 1 行，为通知对象 notification 添加发出提示音效果。因为显示效果是可以叠加的，所以使用“|=”符号表示叠加新的显示效果。如果 notification 的 defaults 字段包括 DEFAULT_SOUND 属性，则这个属性将覆盖 sound 字段中定义的声音。

③ 设置手机振动，如：

```
1  notification.defaults |= Notification.DEFAULT_VIBRATE;
2  long[] vibrate = {0,100,200,300};
3  notification.vibrate = vibrate;
```

✍第 1 行，为通知对象 notification 添加振动效果。

✍第 3 行，设置 0ms 后开始振动，振动 100ms 后停止，再过 200ms 后再次振动 300ms，该设置数据由第 2 行的数组提供。

④ 设置 LED 灯闪烁，如：

```
notification.defaults |= Notification.DEFAULT_LIGHTS;
```

或者，设置自己的 LED 提醒模式：

```
1  notification.ledARGB = 0xff00ff00;
2  notification.ledOnMS = 300;                          //亮的时间
3  notification.ledOffMS = 1000;                        //灭的时间
4  notification.flags |= Notification.FLAG_SHOW_LIGHTS;
```

✍第 1～3 行，设置 LED 灯闪烁的颜色和亮、灭时间。

✍第 4 行，设置 LED 灯按自定义参数的方式进行闪烁。

⑤ 设置对通知的单击事件处理。

```
1  //设置通知的事件消息
2  Context context = getApplicationContext();           //上下文
3  CharSequence contentTitle = "My Notification";       //通知栏标题
4  CharSequence contentText = "Hello World!";           //通知栏内容
5  Intent notificationIntent = new Intent(this,Main.class); //单击该通知后要跳转的 Activity
6  PendingIntent contentIntent = PendingIntent.getActivity(this,0,notificationIntent,0);
7  notification.setLatestEventInfo(context, contentTitle, contentText, contentIntent);
8  //把 Notification 传递给 NotificationManager
9  mNotificationManager.notify(0,notification);
```

✍第 6 行，获取 PendingIntent 对象。PendingIntent 就是一个 Intent 的描述，相当于对 Intent 执行了包装，不一定要马上执行它，只是将其包装后，传递给其他 Activity 或 Application。

✍第 7 行，按第 2～6 行的参数值设置通知在状态栏中的属性。

✍第 9 行，向 NotificationManager 对象发送通知对象 notification。

(4) 发送通知。

```
private static final int ID_NOTIFICATION = 1;
mNotificationManager.notify(ID_NOTIFICATION, notification);
```

9.2 BroadcastReceiver 组件

BroadcastReceiver 类位于 android. content 包下。BroadcastReceiver 顾名思义是广播的接收器,是一种对广播消息进行过滤并响应的控件。它和事件处理机制类似,只不过事件处理机制是程序组件级别的,例如,某个按钮的单击事件,而广播事件处理机制是系统级别的。例如,系统的时区改变、系统时间改变、电池电量低等。到目前为止,只介绍了使用 Intent 来启动一个程序组件,下面将介绍 Intent 的另外一种用法——广播事件。

注意,在程序中实现自己的 BroadcastReceiver 必须要注册。

9.2.1 BroadcastReceiver 的运行机制

BroadcastReceiver 的运行机制比较简单,首先是构建 Intent 对象,然后调用 sendBroadcast()方法将广播发出。当应用程序注册了 BroadcastReceiver 之后,当系统或其他应用程序发送出广播时,所有已经注册的就会检查注册时的 IntentFilter 是否与发送的 Intent 相匹配,若匹配则调用 BroadcastReceiver 的 onReceive()方法,在该方法中响应事件。

1. 发送广播的方式

BroadcastReceiver 有三种发送方式,各种发送方式的不同之处如下。

1) 使用 sendBroadcast()或 sendStickyBroadcast()方法发送的广播

使用 sendBroadcast()或 sendStickyBroadcast()方法发送的广播,所有满足条件的都会执行其 onReceive()方法来处理响应,但若有多个满足条件的 BroadcastReceiver 时,其执行 onReceive()方法的顺序是不定的。

2) 使用 sendOrderedBroadcast()方法发送的广播

使用 sendOrderedBroadcast()方法发送出去的 Intent,会根据 BroadcastReceiver 的注册时 IntentFilter 中设置的优先级顺序地来执行 onReceive()方法,而相同优先级的 BroadcastReceiver,其执行 onReceive()方法的顺序是不定的。

3) 使用 sendStickyBroadcast()方法发送的广播

sendStickyBroadcast()方法与其他两种方法的主要不同是,Intent 在发送后会一直存在,并且在以后调用 registerReceive 注册相匹配的 Receive 时会把这个 Intent 直接返回给新注册的 Receive。

2. 接收广播的过程

BroadcastReceiver 的接收程序开发过程如下。

1) 开发 BroadcastReceiver 类的子类,并重写 onReceive()方法

继承 BroadcastReceiver 类,并重写 onReceive()方法的代码段如下:

```
public class MyReceiver extends BroadcaseReceiver {
    @Override
    Public void onReceive(Context context, Intent intent) {
    //…… 在这里定义事件响应处理
    }
}
```

在 onReceive()方法里最好不要有执行超过 5s 的代码，如果这样，Android 系统就会弹出一个超时对话框。因此，建议把比较耗时的事件响应方法写在一个线程里，单独来执行。

2）注册 BroadcastReceiver 对象

不管是系统广播的 Intent 还是其他程序广播的 Intent，如果想接收并且对它进行处理，都需要注册一个 BroadcastReceiver，并且一般地要给注册的这个 BroadcastReceiver 设置一个 IntentFilter 来制定当前的 BroadcastReceiver 是对哪些 Intent 进行监听。注册 BroadcastReceiver 有以下两种方式。

（1）静态注册方式。

静态注册方式是在 AndroidManifest.xml 的<application>标签里添加<receiver>标签，并设置要接收的 action。这种方式比较常用。例如对接收系统日期改变完成广播的注册，其注册代码为：

```
<receiver android:name = ".MyReceiver">
    <intent - filter>
        <action android:name = "android.intent.action.DATE_CHANGED"/>
        <category android:name = "android.intent.category.HOME"/>
    </intent - filter>
</receiver>
```

Android 系统提供了很多标准广播 Action。这些广播是系统自动发出的，可以在<intent-filter>的<action>中直接使用这些标准广播常量。常见的标准广播 Action 常量如表 9-3 所示。

表 9-3　常见的标准广播 Action 常量

常　量　名	常　量　值	含　义
ACTION_BOOT_COMPLETED	Android.intent.action.BOOT_COMPLETED	系统启动完成
ACTION_TIME_CHANGED	Android.intent.action.ACTION_TIME_CHANGED	时间改变
ACTION_DATE_CHANGED	Android.intent.action.ACTION_DATE_CHANGED	日期改变
ACTION_TIMEZONE_CHANGED	Android.intent.action.ACTION_TIMEZONE_CHANGED	时区改变
ACTION_BATTERY_LOW	Android.intent.action.ACTION_BATTERY_LOW	电量低
ACTION_MEDIA_EJECT	Android.intent.action.ACTION_MEDIA_EJECT	插入或拔出外部媒体

（2）动态注册方式。

动态注册方式是在 Activity 里通过调用 registerReceiver()方法来注册。例如，同样地，要完成接收系统日期改变广播，使用在代码中直接注册，其代码如下所示。

```
……
    //创建相关对象
    MyReceiver receiver = new MyReceiver();
    IntentFilter filter = new IntentFilter();
    filter.addAction(DATE_CHANGED);

    //动态注册 BroadcastReceiver
    registerReceiver(receiver, filter);
……
```

registerReceiver()方法有两个参数，第一个参数是指定接收器对象，第二个参数是 IntentFilter 对象，其内是要接收的 action 属性。

(3) 两种注册方式比较。

现在已经了解了两种注册 BroadcastReceiver 的方式，还需要了解两种注册方式的特点，以便在开发中可根据应用需求的场合来选择注册方式。

静态注册方式的特点：在应用程序安装之后，无论该应用程序是否处于活动状态，BroadcastReceiver 始终处于被监听状态。这种注册方式通常用于监听系统状态的改变，如手机的电量，日期的改变，时期的改变，WIFI 网卡的状态等。

动态注册方式的特点：在代码中进行注册后，当应用程序关闭后，就不再进行监听。这样注册的 BroadcastReceiver 通常用于更新 UI 的状态。一般情况下，在 Activity 的 onResume() 方法中使用 Context.registerReceiver()方法来注册一个广播接收器，在 onPause()方法中使用 Context.unregisterReceiver(r)方法来注销一个广播接收器。

注册的 BroadcastReceiver 并非一直在后台运行，一旦当事件或相关的 Intent 传来，就会被系统调用，处理 onReceive()方法里的响应事件。

9.2.2　BroadcastReceiver 的应用案例

BroadcastReceiver 组件没有提供可视化的界面来显示广播信息，但可以使用 Notification 来实现广播信息的显示、提示音、闪烁和振动等消息提示。下面是一个使用 BroadcastReceiver 和 Notification、NotificationManager 的综合案例。

【案例 9.2】 通过一个按钮来发出一个广播信息，并将它显示在状态栏中。也可以单击一个按钮，清除状态栏中的通知信息。

【说明】 这是用户自己定义的 BroadcastReceiver 发送广播的案例，而非系统级的广播，所以在程序代码中使用 registerReceiver()方法动态地注册 BroadcastReceiver 的方式。

本例应用需要定义两个程序类，一个是 Activity 类，在其中构建案例中需要广播的 Intent，然后在按钮的 onClick()方法中使用 sendBroadcast()方法发送出去，并且在该类的 onResume()方法中注册广播接收器。另一个是继承 BroadcastReceiver 的子类，在该子类的 onReceive()方法中将传入的广播 Intent 中的信息发送给一个通知对象。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 BroadcastRec_Demo 的 Android 项目。其应用程序名为 BroadcastRec，包名为 cn.com.sgmsc.broadcastreceiver，Activity 组件名为 BroadcastRec_DemoActivity。

(2) 准备图片。将用于通知的图标的图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 准备字符串资源。编写 res/values 目录下的 strings.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>

    <string name = "hello">Hello World, BroadcastRec_DemoActivity!</string>
    <string name = "app_name">BroadcastReceiver</string>
    <string name = "btnsend">发送广播</string>
    <string name = "btnclr">清除通知</string>
    <string name = "brdcast">这是 BroadcastReceiver 发出的广播。</string>

</resources>
```

(4) 设计布局。编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent" >
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/app_name"
    android:gravity = "center_horizontal"
    android:padding = "10sp"
    android:textSize = "8pt"
    android:textColor = "#00FF00" />
<Button android:id = "@ + id/BTN_SEND"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_gravity = "center_horizontal"
    android:text = "@string/btnsend" />
<Button android:id = "@ + id/BTN_CLEAR_NOTIFICATION"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_gravity = "center_horizontal"
    android:text = "@string/btnclr" />
</LinearLayout>
```

两个按钮显示的文本取自于 strings.xml 文件。

(5) 开发逻辑代码。打开 src/cn.com.sgmsc.broadcastreceiver 包下的 BroadcastRec_DemoActivity.java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.broadcastreceiver;

import android.app.Activity;
import android.app.NotificationManager;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class BroadcastRec_DemoActivity extends Activity implements OnClickListener {

  public static final String ACTION = "cn.com.sgmsc.action.BroadcastReceiver";
                                                              //定义 IntentFilter 的 action
  public static final String INTENT_EXTRAS_TITLE = "BroadcastReciver Demo Activity.";

  private MyReceiver mReceiver = null;
  private IntentFilter mIntentFilter = null;

  @Override
  protected void onCreate(Bundle icicle) {
      super.onCreate(icicle);
      setContentView(R.layout.main);
```

```
24
25          Button btnSend = (Button)findViewById(R.id.BTN_SEND);   //"发送广播"按钮
26          Button btnClear = (Button)findViewById(R.id.BTN_CLEAR_NOTIFICATION);
                                                                    //"清除通知"按钮
27          btnSend.setOnClickListener(this);
28          btnClear.setOnClickListener(this);
29      }
30
31      @Override
32      protected void onResume() {
33          mReceiver = new MyReceiver();
34          mIntentFilter = new IntentFilter();
35          mIntentFilter.addAction(ACTION);                //向 mIntentFilter 中添加 action 属性值
36          this.registerReceiver(mReceiver, mIntentFilter);        //注册接收器
37          super.onResume();
38      }
39
40      @Override
41      protected void onDestroy() {
42          if(mReceiver != null) {                                 //注销接收器
43              this.unregisterReceiver(mReceiver);
44          };
45          super.onDestroy();
46      }
47
48      @Override
49      public void onClick(View v) {
50          switch(v.getId() ) {
51              case R.id.BTN_SEND: {
52                  doSend();
53                  break;
54              }
55              case R.id.BTN_CLEAR_NOTIFICATION: {
56                  doclearnotification();
57                  break;
58              }
59          }
60      }
61
62      private void doSend() {
63          Intent sendIntent = new Intent(ACTION);
64          sendIntent.putExtra("extit", INTENT_EXTRAS_TITLE);
65          sendIntent.putExtra("exmsg",getString(R.string.brdcast).toString());
66          this.sendBroadcast(sendIntent);  //广播发送一个 Intent 对象 sendIntent
67      }
68
69      private void doclearnotification() {
70          NotificationManager notificationManager = (NotificationManager)
71                      this.getSystemService(android.content.Context.NOTIFICATION_SERVICE);
72            notificationManager.cancel(MyReceiver.NOTIFICATION_ID);
                                                    //清除 ID 号为 NOTIFICATION_ID 的通知
73      }
74
75  }
```

① 第 12 行定义这个 Activity 类实现 OnClickListener 接口。在该接口内重定义了

onClick()方法，当单击“发送广播”按钮时执行 doSend()方法，该方法在第 62～67 行定义。当单击“清除通知”按钮时执行 doclearnotification()方法，该方法在第 69～73 行定义。

② 第 32～38 行重写了 onResume()方法，在该方法中使用 registerReceiver()方法注册广播接收器。

③ 第 41～46 行重写了 onDestroy()方法，在该方法中使用 unregisterReceiver()方法注销了广播接收器。

(6) 开发 MyReceiver 类代码。在 src/cn. com. sgmsc. broadcastreceiver 包下创建一个代码文件 MyReceiver. java，该文件是定义一个广播接收器类 MyReceiver，它继承自 BroadcastReceiver 类，并且重写 onReceiver()方法，在该方法内编写响应接收广播的事件。其代码如下所示。

```
package cn.com.sgmsc.broadcastreceiver;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class MyReceiver extends BroadcastReceiver {
 Context context;
 public static int NOTIFICATION_ID = 21321;

 @Override
 public void onReceive(Context context, Intent intent) {
     this.context = context;
         //从传入的 Intent 对象中取出该 Intent 附加的信息
     String titstr = intent.getStringExtra("extit");
     String msgstr = intent.getStringExtra("exmsg");

         //获得 NotificationManager 的实例
     NotificationManager notificationmanager = (NotificationManager) context
             .getSystemService(android.content.Context.NOTIFICATION_SERVICE);
     //实例化 Notification
     Notification notification = new Notification(R.drawable.ic_header,
             titstr, System.currentTimeMillis());
         //获得 PendingIntent 对象
     PendingIntent pintent = PendingIntent.getActivity(context, 0,
                             new Intent(context, BroadcastRec_DemoActivity.class), 0);
         //设置事件信息
     notification.setLatestEventInfo(context, msgstr, null,pintent);
     //发出 ID 号为 NOTIFICATION_ID 的通知
     notificationmanager.notify(NOTIFICATION_ID, notification);
 }

}
```

(7) 静态注册广播接收器的代码。本例是在代码中动态注册 BroadcastReceiver 的。如果想使用静态注册的方法，则在 AndroidManifest. xml 文件中声明。具体的代码见该 AndroidManifest. xml 文件的第 21～26 行的注释部分。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "cn.com.sgmsc.broadcastreceiver"
    android:versionCode = "1"
    android:versionName = "1.0" >

    < uses - sdk android:minSdkVersion = "10" />

    < application
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name" >
        < activity
            android:label = "@string/app_name"
            android:name = ".BroadcastRec_DemoActivity" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity>

    <! -- receiver android:name = ".MyReceiver">
        < intent - filter >
            < action
                android:name = "cn.com.sgmsc.action.BroadcastReciver"/>
        </intent - filter>
    </receiver -->

  </application>

</manifest>
```

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 BroadcastRec_Demo 项目。初始运行的显示效果如图 9-4 所示。当单击"发送广播"按钮后，自定义的广播接收器接收广播的 Intent，并将传入的信息显示在状态条中，如图 9-5 所示。当拉下状态条，出现状态栏信息，如图 9-6 所示。当单击"清除通知"按钮后，状态条中的信息被清除，如图 9-7 所示。



图 9-4　初始运行界面



图 9-5　单击"发送广播"按钮后



图 9-6 拉下状态条时显示的通知信息



图 9-7 单击“清除通知”按钮后

9.3 Android 后台线程

当一个程序第一次启动时，Android 会启动一个 Linux 进程和一个主线程(Main Thread)。线程是比进程更小的执行单位。

主线程主要负责处理与 UI 相关的事件，如用户的按键事件、用户接触屏幕的事件以及屏幕绘图事件，等等，并把相关的事件分发到对应的组件进行处理。所以主线程通常又被叫作 UI 线程。在开发 Android 应用时必须遵守单线程(Single-threaded)模型的原则：Android UI 操作并不是线程安全的，并且这些操作必须在 UI 线程中执行。

所有非主线程就是子线程，子线程一般都是后台线程，它是由用户在程序中定义的。由于 Android 的 UI 是单线程的，为了保证应用程序的响应效率，一般使用后台线程，把所有运行慢的、耗时的操作移出主线程，放到子线程中。例如，Activity 在 5s 内不响应任何输入事件，或者 Broadcast Receiver 在 5s 后仍未完成 onReceive 的处理操作，等等。

在子线程中操作 UI 对象是不安全的。如果后台的子线程执行了 UI 对象，Android 就会发出错误信息 CalledFromWrongThreadException。为了解决后台子线程与 UI 线程间的信息交互问题，Android 设计了一种 Handler 消息传递机制或 AsyncTask 后台运行事务，来处理线程之间的数据传递问题。

9.3.1 Handler 消息传递机制

1. Handler 类

Handler 类位于 android.os 包下。它负责消息(Message)的发送，消息内容的执行处理，主要完成 Android 的 Widget 与应用程序中子线程之间的交互。自定义的后台线程可与 Handler 通信，一个 Handler 对应一个 Activity，Handler 将与 UI 线程一起工作。

消息(Message)可以理解为线程间交流的信息。消息队列(Message Queue)则是用来存放通过 Handler 发布的消息，每个消息队列都会有一个对应的 Handler，按照先进先出执行。

Handler 的常用方法如表 9-4 所示。

表 9-4　Handler 中的常用方法及说明

方　法	返回值	描　述
handlerMessage(Message msg)	void	子类对象通过该方法接收消息
sendEmptyMessage(int what)	boolean	发送一个只含有 what 值的消息
sendMessage(Message msg)	boolean	发送消息到 Handler
hasMessages(int what)	boolean	监测消息队列中是否还有 what 值的消息
Post(Runnable r)	boolean	将一个线程添加到消息队列中

2. Handler 类的使用

开发带有 Handler 类的程序步骤如下。

(1) 在 Activity 或 Activity 的 Widget 中开发 Handler 类的对象，重写 handlerMessage()方法。

(2) 在新启动的线程中调用 sendEmptyMessage()或 sendMessage()方法向 Handler 发送消息。

(3) Handler 类的对象用 handlerMessage()方法接收消息，然后根据消息执行相应的操作。

下面通过一个 Handler 使用案例来介绍其用法。

【案例 9.3】 使用 Handler 机制以幻灯片的形式显示 IT 界的名人照片，每张图片停留 3s。

【说明】 本例使用两个类来定义代码，一个是 Activity 类，在其中构建 Handler 对象，并重写 handlerMessage()方法，在该方法内根据消息的值确定 UI 的显示内容。另一个是继承 Thread 的子类，并重写 run()方法，在该方法内使用 sendEmptyMessage()方法向 Handler 发送消息。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 ThreadDemo 的 Android 项目。其应用程序名为 ThreadDemo，包名为 cn.com.sgmsc.threaddemo，Activity 组件名为 ThreadDemoActivity。

(2) 准备图片。将要显示的图片资源复制到本项目的 res/drawable-mdpi 目录中。

(3) 准备字符串资源。编写 res/values 目录下的 strings.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>

    <string name = "hello">Hello World, ThreadDemoActivity!</string>
    <string name = "app_name">ThreadDemo</string>

    <string name = "title">IT 界名人</string>
    <string name = "andy">Andy Rubin,Android 的创始人之一。</string>
    <string name = "bill">Bill Joy,Java 创造者之一。</string>
    <string name = "edgar">Edgar F. Codd,关系数据库之父。</string>
    <string name = "torvalds">Linus Torvalds,Linux 系统的创始人。</string>
    <string name = "turing">Turing Alan,IT 的祖师爷。</string>
</resources>
```

(4) 设计布局。编写 res/layout 目录下的 main. xml 文件。在这里使用 RelativeLayout 布局,代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout
        xmlns:android = "http://schemas.android.com/apk/res/android"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_margin = "2dip">

    < TextView
        android:id = "@ + id/picTitle"
        android:layout_width = "wrap_content"
        android:layout_height = "50dip"
        android:layout_alignParentTop = "true"
        android:layout_centerHorizontal = "true"
        android:gravity = "center_vertical"
        android:textSize = "22sp"
        android:textColor = " #ffffff00"
        android:text = "@string/title" />

    < ImageView
     android:id = "@ + id/myPic"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_below = "@id/picTitle"
        android:layout_alignParentTop = "true"
        android:layout_alignParentLeft = "true"
        android:src = "@drawable/andy" />

    < TextView
        android:id = "@ + id/picName"
        android:layout_width = "fill_parent"
        android:layout_height = "50dip"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentLeft = "true"
        android:gravity = "center"
        android:background = " #55ffff00"
        android:textSize = "18sp"
        android:text = "@string/andy" />

</RelativeLayout >
```

(5) 开发逻辑代码。打开 src/cn. com. sgmsc. threaddemo 包下的 ThreadDemoActivity. java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.threaddemo;

import android.os.Handler;
import android.os.Message;
//…… 省略了部分引入类

public class ThreadDemoActivity extends Activity {
 ImageView myPicture;                                    //ImageView 的引用
 TextView myPicname;                                     //TextView 的引用
```

```
10 Handler myHandler = new Handler(){                           //创建一个 Handler 对象
11     @Override
12     public void handleMessage(Message msg) {                 //重写接收消息的方法
13         switch(msg.what){                                    //判断 what 的值
14         case 0:                                              //what 值为 0 时
15             myPicture.setImageResource(R.drawable.andy);
16             myPicname.setText(R.string.andy);
17             break;
18         case 1:                                              //what 值为 1 时
19             myPicture.setImageResource(R.drawable.bill);
20             myPicname.setText(R.string.bill);
21             break;
22         case 2:                                              //what 值为 2 时
23             myPicture.setImageResource(R.drawable.edgar);
24             myPicname.setText(R.string.edgar);
25             break;
26         case 3:                                              //what 值为 3 时
27             myPicture.setImageResource(R.drawable.torvalds);
28             myPicname.setText(R.string.torvalds);
29             break;
30         case 4:                                              //what 值为 4 时
31             myPicture.setImageResource(R.drawable.turing);
32             myPicname.setText(R.string.turing);
33             break;
34         }
35         super.handleMessage(msg);
36     }
37     };
38     @Override
39     public void onCreate(Bundle savedInstanceState) {        //重写的 onCreate()方法
40         super.onCreate(savedInstanceState);
41         setContentView(R.layout.main);                       //设置当前的用户界面
42         myPicture = (ImageView) findViewById(R.id.myPic);    //得到 ImageView 的引用
43         myPicname = (TextView) findViewById(R.id.picName);   //得到 ImageView 的引用
44         MyThread myThread = new MyThread(this);              //初始化 MyThread 线程
45         myThread.start();                                    //启动线程
46     }
47
48 }
```

① 第 10～37 行创建一个 Handler 对象 myHandler，并重写了 handleMessage()方法，在该方法中根据传入的 message 的 what 值，确定 ImageView 和 TextView 中的显示内容。

② 第 45 行，在 onCreate()方法中启动线程。

(6) 开发 MyThread 类代码。在 src/cn.com.sgmsc.threaddemo 包下创建一个代码文件 MyThread.java，该文件是定义一个线程类 MyThread，它继承自 Thread 类，并且重写 run()方法，其代码如下所示。

```
package cn.com.sgmsc.threaddemo;

public class MyThread extends Thread{
  ThreadDemoActivity activity;                                  //Activity 的引用
  int what = 1;                                                 //发送消息的 what 值
  public MyThread(ThreadDemoActivity activity){                 //构造器
      this.activity = activity;                                 //得到 Activity 的引用
```

```
8       }
9       @Override
10      public void run() {                                    //重写的 run()方法
11          while(true){                                       //循环
12              activity.myHandler.sendEmptyMessage((what++) % 5);  //发送消息
13              try{
14                  Thread.sleep(3000);                        //睡眠
15              }
16              catch(Exception e){                            //捕获异常
17                  e.printStackTrace();                       //打印异常信息
18              }
19          }
20      }
21  }
```

第 10～20 行，在 run()方法中执行一个循环，使用 sendEmptyMessage()方法每隔 3s 向 Activity 的 myHandler 中发送一个消息值，该值由表达式(what＋＋)%5 计算，其计算结果是 0，1，2，3，4。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 ThreadDemo 项目。运行结果是以 3s 的时间间隔逐一显示资源中提供的人物照片，并且循环往复。其显示效果如图 9-8 和图 9-9 所示。



图 9-8 显示的第一张照片



图 9-9 3s 后显示的第二张照片

当然，也可以在同一个 Activity 中定义子线程。在第 6 章案例 6.3 中，就是这样实现线程通信的。下面来重温其相关代码。在 Eclipse 中展开项目 Activity_Dialog，打开 cn.com.sgmsc.dialog 包下的 DialogActivity.java 文件，与子线程相关的代码片断如下。

```
package cn.com.sgmsc.dialog;                          //声明包语句

//…… 省略了部分引入类
import android.os.Handler;                            //引入相关类
import android.os.Message;                            //引入相关类

public class DialogActivity extends Activity {
    //…… 省略部分对象、变量的引用声明
```

```
9       ProgressDialog pd;
10      Handler myHandler;                                    //Handler 对象引用
11
12      @Override
13      public void onCreate(Bundle savedInstanceState) {
14          //…… 省略部分初始化代码
15          myHandler = new Handler(){                        //创建 Handler 对象
16              @Override
17              public void handleMessage(Message msg){
18                  switch(msg.what){
19                      case INCREASE:
20                          pd.incrementProgressBy(1);        //进度每次加 1
21                              if(pd.getProgress()>= 100){   //判断是否结束进度
22                                  pd.dismiss();             //如果进度条走完则关闭窗口
23                              }
24                      break;
25                  }
26                  super.handleMessage(msg);
27              }
28          };
29      }
30
31      //…… 省略部分代码段
32
33      @Override
34      public void onPrepareDialog(int id, Dialog dialog){
                                          //每次弹出对话框时更新对话框内容的方法
35        super.onPrepareDialog(id, dialog);
36        switch(id){
37              case  PROGRESS_DIALOG:
38                  pd.incrementProgressBy( - pd.getProgress());    //对话框进度清零
39                  new Thread(){                                   //创建一个线程
40                  public void run(){
41                      while(true){
42                          myHandler.sendEmptyMessage(INCREASE);   //发送 Handler 消息
43                          if(pd.getProgress()>= 100){
44                              break;
45                          }
46                          try{
47                              Thread.sleep(40);                   //线程休眠
48                          }
49                          catch(Exception e){
50                              e.printStackTrace();                //捕获并打印异常
51                          }
52                      }
53                  }
54              }.start();                                          //启动线程
55              break;
56      }
57
58 }
```

① 第 10～20 行，在 onCreate()方法中创建一个 Handler 对象 myHandler，并根据消息指令，如果消息为 INCREASE，那么在进度条未到达终点时就让进度条前进一格进度，否则退出

进度条对话框。

② 第39～54行创建了一个匿名线程，在该线内重写了run()方法。第54行，该线程一经创建即刻启动。

看上去，这种匿名线程定义简单，不需要另外定义继承子类。但是这种匿名线程的方式是存在缺陷的：第一，线程的开销较大，如果每个任务都要创建一个线程，那么应用程序的效率要低很多；第二，线程无法管理，匿名线程创建并启动后就不受程序的控制了，如果有很多个请求发送，那么就会启动非常多的线程，系统将不堪重负。另外，前面已经看到，在新线程中更新UI还必须要引入Handler，这让代码看上去非常臃肿。为了解决这一问题，Android引入了AsyncTask。

9.3.2 AsyncTask

1. AsyncTask类

AsyncTask是抽象类，位于android.os包下。顾名思义，AsyncTask是异步任务的意思，它是一种简单的实现后台运行事务的方式。异步任务是Android 1.5引入的一个新特性，UI线程可以把某些操作留给后台运行的线程去运行，Android自动新建和删除后台线程，无须开发者显式地实现。所以，AsyncTask的特点是任务在UI线程之外运行，而回调方法是在UI线程中执行，这就有效地避免了使用Handler带来的麻烦。使用AsyncTask类可以将耗时的操作放在后台来处理，而不需要人为地另开线程或者使用Handlers来完成。

AsyncTask是一个抽象类，要使用这个类则必须通过继承该类的子类，重写其中的方法。下面是这个类中常常需要重写的一些方法，如表9-5所示。

表9-5　AsyncTask类需要重写的方法及说明

方　法	描　述
onPreExecute()	当任务执行之前开始调用此方法，可以在这里显示进度对话框
doInBackground(Params...)	此方法在后台线程执行，完成任务的主要工作，通常需要较长的时间。在执行过程中可以调用
publicProgress(Progress...)	更新任务的进度
onProgressUpdate(Progress...)	此方法在主线程执行，用于显示任务执行的进度
onPostExecute(Result)	此方法在主线程执行，任务执行的结果作为此方法的参数返回

开发时，在AsyncTask子类中必须实现抽象方法doInBackground(Params… p)，这个方法是必须要重写的，用来在后台线程中处理一些耗费时间的事情，当这个方法执行完毕后会返回一个值，这个值是onPostExecute(Object res)方法的参数，也就是说doInBackground()方法运行结束后传递参数给onPostExecute()交由其执行。当然在doInBackground()中可以调用publishProgress()方法传递参数，这些值将会在onProgressUpdate()方法中来实现UI线程中控件信息的更新。

AsyncTask中有三个参数，例如class MyTask extends AsyncTask< Params,Progress,Result >{}。这三个参数是一种范式类型，定义这种范式参数的格式为"<数据类型>…<参数名>"。它们被用于一个异步任务。在一个异步任务里，不是所有的类型总被用。假如一个类型不被使用，可以简单地使用Void类型来替代，例如doInBackground(Void…Params)。这里，三个参数的作用如下。

(1) Params：当执行时，向后台任务的执行方法传递参数的类型，例如 HTTP 请求的 URL。

(2) Progress：在后台任务执行过程中，要求主 UI 线程处理中间状态，通常是一些 UI 处理中传递的参数类型，例如在后台计算期间，后台任务执行的百分比。

(3) Result：后台任务执行完返回时的参数类型，后台计算的结果类型，例如 String。

AsyncTask 类除了上述需要重写的方法外，还有一些常用的公共方法，如表 9-6 所示。

表 9-6　AsyncTask 中常用的方法及说明

方法	返回值	描　述
cancel (boolean mayInterruptIfRunning)	boolean	尝试取消这个任务的执行。如果此任务不能取消返回 false，如果已经正常执行完毕，返回 true
execute (Params...params)	AsyncTask	用指定的参数来执行此任务，这个方法将会返回此任务本身。此方法必须在 UI 线程中调用
get (long timeout，TimeUnit unit)	Result	等待计算结束并返回结果，最长等待时间为：timeout (超时时间)
getStatus ()	AsyncTask. Status	获得任务的当前状态
isCancelled ()	boolean	如果在任务正常结束之前取消任务成功则返回 true，否则返回 false

2. AsyncTask 的使用步骤

AsyncTask 的执行分为若干步骤，每一步都对应一个回调方法，这些方法需要重写。注意：在任务的执行过程中，这些方法不由应用程序调用，而是被自动调用的。

一个 AsyncTask 运行的过程中，经历了以下 4 个步骤。

1) onPreExecute()

在 execute 调用后立即在 UI 线程中执行。这步通常被用于设置任务，例如在用户界面显示一个进度条。

2) doInBackground(Params...)

当 onPreExecute()执行完成后，立即在后台线程中运行。这步被用于执行较长时间的后台计算。异步任务的参数也被传到这步。计算的结果必须在这步返回，将传回到上一步。在执行过程中可以调用 publishProgress(Progress...)来更新任务的进度。

3) onProgressUpdate(Progress...)

在调用 publishProgress 后，在 UI 线程中运行。执行的时机是不确定的。这个方法用于当后台计算还在进行时在用户界面显示进度。例如：这个方法可以被用于一个进度条动画或在文本域显示日志。

4) onPostExecute(Result)

当后台计算结束时，调用 UI 线程。后台计算结果作为一个参数传递到这步。

3. AsyncTask 的使用规则

因为 AsyncTask 的特殊性，很多开发工作都被省略了，如下工作是不需要开发者来完成的：产生自己的后台线程；在适当的时候终止后台线程等。AsyncTask 具有 fire-and-forget (发射后自寻)的特性，就是线程产生后不用用户干涉，全部自发完成。所以在编程中，必须遵

守一些线程规则，AsyncTask 类才能正确工作。具体的规则如下。

(1) AsyncTask 任务实例必须创建在 UI 线程中。

(2) execute(Params...)必须在 UI 线程上调用。

(3) 不要手动调用 onPreExecute()、onPostExecute(Result)、doInBackground(Params...)、onProgressUpdate(Progress...)方法。

(4) 每个 AsyncTask 只能有一个实例被执行，同时运行两个以上的 AsyncTask，将会抛出异常。

4. AsyncTask 的应用

下面通过一个简单的例子来说明使用 AsyncTask 完成后台操作任务的整个过程。

【案例 9.4】 使用 AsyncTask 来完成进度对话框的进度更新操作。

【说明】 由于使用 AsyncTask，任务在后台运行，在前台界面是看不到后台执行过程的，本例在代码中增加了输出 Log 的操作，便于读者看到其执行过程。

问一下：

怎样实现输出 Log?

使用 android. util 的 Log 类可以实现 Android 输出 Log 这一操作。android. util. Log 常用的方法有以下 5 个：Log. v(String tag,String text)，Log. d(String tag,String text)，Log. i(String tag,String text)，Log. w(String tag,String text)以及 Log. e(String tag,String text)。根据首字母对应 VERBOSE,DEBUG,INFO,WARN,ERROR。

不同的方法在 logCat 里面显示文本的颜色不同。Log. v 的输出颜色为黑色的，任何消息都会输出；Log. d 的输出颜色是蓝色的，仅输出调试信息；Log. i 的输出为绿色，是一般提示性的消息；Log. w 的输出为橙色，是一些警告信息；Log. e 为红色，是一些错误信息。

在 Eclipse 的控制台的 LogCat 窗口中可以看到 Log 信息。调出 LogCat 窗口的方法是：选择菜单 Window→Show View→LogCat，当单击 Ok 按钮时，会在控制台窗口中出现 LogCat 窗口。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 AsyncTask_PgDialog 的 Android 项目。其应用程序名为 ProgressDialog_AT，包名为 cn. com. sgmsc. asynctask_PGD，Activity 组件名为 AsyncTask_PgDialogActivity。

(2) 设计布局。编写 res/layout 目录下的 main. xml 文件。其代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal"
    >

    <Button
        android:text = "显示进度对话框"
        android:id = "@ + id/Button01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
```

```
        />

</LinearLayout>
```

(3) 开发逻辑代码。打开 src/cn. com. sgmsc. asynctask_PGD 包下的 AsyncTask_PgDialogActivity. java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.asynctask_PGD;

import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.SystemClock;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.app.Dialog;
import android.app.ProgressDialog;

public class AsyncTask_PgDialogActivity extends Activity {
  ProgressDialog pd;

  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //打开进度对话框的按钮
    Button bok = (Button)this.findViewById(R.id.Button01);//获得 Button 对象
    bok.setOnClickListener(                           //设置 OnClickListener 监听器
      new View.OnClickListener(){
      @Override
      public void onClick(View v) {                   //重写 onClick()方法
        showDialog(1);                                //显示进度对话框

        //步骤 1: 创建后台任务的对象,并通过 execute()启动后台线程,
        //调用 doInBackground()的代码,execute 中的参数类型为参数 1,这里不需要传递任何内容
        new AddStringTask().execute ();
      }
    }
   );
  }

  @Override
  protected Dialog onCreateDialog(int id) {           //重写 onCreateDialog()方法
    pd = new ProgressDialog(this);                    //创建进度对话框
    pd.setMax(100);                                   //设置最大值
    pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);//水平进度条
    pd.setTitle("完成进度");                           //设置标题
    pd.setMessage("请稍等...");
    pd.setCancelable(false);     //设置进度对话框不能用“回退”按钮关闭

    return pd;                                        //返回生成 Dialog 的对象

  }
```

```
49
50  //步骤 2: 创建 AsyncTask 子类,参数 1 是 Void 的范式类型,参数 2 是 Integer 的范式类型,参数 3
    //是 Void。
51  private class AddStringTask extends AsyncTask < Void, Integer, Void > {
52    //这里加入一个检测信息的方法,打印当前在哪个线程执行的信息
53    private void printInfo(String info){
54      Log.d("SGMSC", info + " : Tread is " + Thread.currentThread().getName());
55    }
56
57    //步骤 3: 实现抽象方法 doInBackground(),代码将在后台线程中执行,由 execute()触发
58    protected Void/ * 参数 3 * / doInBackground (Void... params/ * 参数 1 * /) {
59      pd.incrementProgressBy( - pd.getProgress());        //对话框进度清零
60      for(int i = 1;i <= 100;i++){
61        //步骤 2: 通知 UI 主线程执行相关的操作(在 onProgressUpdate 中定义)
62        publishProgress( i/ * 参数 2 * / );
63        printInfo("doInBackgound " + i);
64        SystemClock.sleep(50);
65      }
66      return (null);
67    }
68
69    //步骤 3: 定义收到 publishProgress()触发后,在 UI 主线程执行的内容
70    protected void onProgressUpdate (Integer... values/ * 参数 2 * /) {
71      printInfo("onProgressUpdate  get param " + values[0]);
72
73      pd.incrementProgressBy(1);                           //进度每次加 1
74      if(values[0]>= 100){                                 //判断是否结束进度
75          pd.dismiss();                                    //如果进度条走完则关闭窗口
76      }
77    }
78
79    //步骤 4: 定义后台进程执行完后的处理,本例采用 Toast
80    protected void onPostExecute (Void result/ * 参数 3 * / ) {
81      printInfo("onPostExecute");
82      Toast.makeText(AsyncTask_PgDialogActivity.this, "It's Over!", Toast.LENGTH_SHORT).show();
83    }
84  }
85 }
```

① 第 31 行,是使用 AsyncTask 的第一步。在 onCreate()方法中调用自定义类 AddStringTask 的 execute()方法,创建后台任务的对象,并通过 execute()启动后台线程。在后面定义子类 AddStringTask 中调用 doInBackground()的代码,execute 中的参数类型为 AsyncTask 的参数 1,这里不需要传递任何内容。

② 第 51～84 行定义了子类 AddStringTask。其中,参数 1 是 Void 的范式类型,是向后台任务的执行方法传递参数的类型;参数 2 是 Integer 的范式类型,在后台任务执行过程中,传递给主 UI 线程中进度条的进度状态参数;参数 3 是 Void,是后台任务执行完返回时的参数类型。

③ 第 54 行,创建一个私有方法,该方法是将操作信息 info 输出到 Log 中,该 info 在重写方法 doInBackground()中提供。

④ 第 58～67 行,实现抽象方法 doInBackground(),代码将在后台线程中执行,由 execute()触发。由于这个例子并不需要传递参数,所以使用"Void..."的范式书写方式。在后台中首先

执行将进度条清零操作，然后使用从 1 到 100 的循环，每一步循环作下列操作：调用 publishProgress(i)以更新主 UI 线程中进度对话框的状态，向 Log 输出信息，系统休眠 0.05s。

⑤ 第 70～77 行，定义收到 publishProgress()触发后，在 UI 主线程执行的内容。在本例，将进度数 i 传入 values 中。其中的参数为范式方式，实质为 Integer 数组，即传到 values[0]中。第 71 行向 Log 输出信息；第 73～76 行，将进度条的进度进 1，当进度条达到 100 时关闭进度条对话框。

⑥ 第 80～83 行重写 onPostExecute()方法，定义后台进程执行完后的处理，向 Log 输出的 text 部分的信息为“onPostExecute”，并采用 Toast 向用户界面显示提示消息。

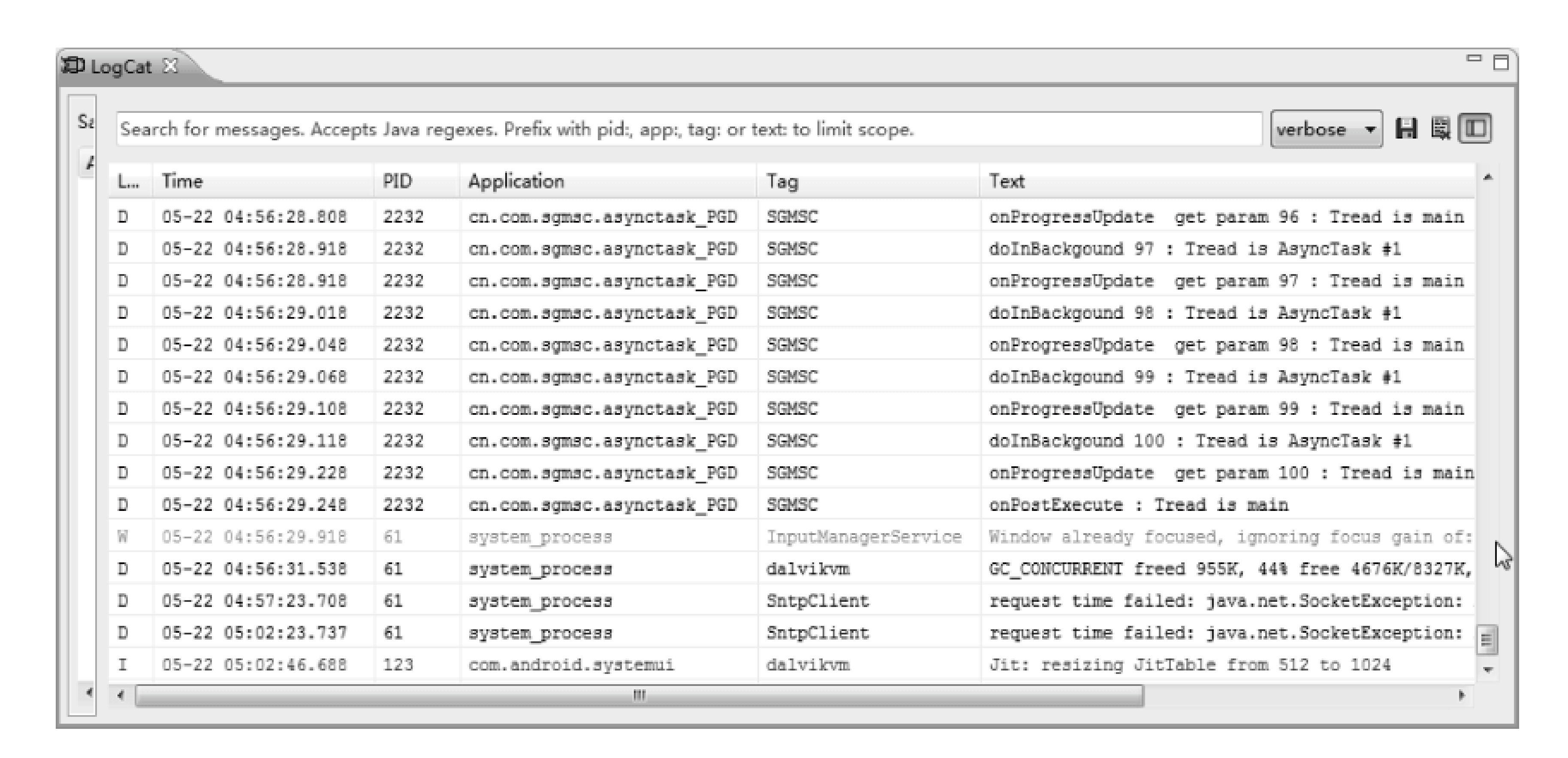
【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 AsyncTask_PgDialog 项目。初始运行界面如图 9-10 所示，单击“显示进度对话框”按钮后，随即出现进度对话框，且进度条的刻度逐渐向前递进，直到 100%结束，如图 9-11 所示。然后，可以在 Eclipse 的 LogCat 窗口中看到后台操作过程的日志记录信息，如图 9-12 所示。



图 9-10 初始运行项目的界面



图 9-11 单击按钮之后调出进度对话框



L...	Time	PID	Application	Tag	Text
D	05-22 04:56:28.808	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	onProgressUpdate get param 96 : Tread is main
D	05-22 04:56:28.918	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	doInBackgound 97 : Tread is AsyncTask #1
D	05-22 04:56:28.918	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	onProgressUpdate get param 97 : Tread is main
D	05-22 04:56:29.018	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	doInBackgound 98 : Tread is AsyncTask #1
D	05-22 04:56:29.048	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	onProgressUpdate get param 98 : Tread is main
D	05-22 04:56:29.068	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	doInBackgound 99 : Tread is AsyncTask #1
D	05-22 04:56:29.108	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	onProgressUpdate get param 99 : Tread is main
D	05-22 04:56:29.118	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	doInBackgound 100 : Tread is AsyncTask #1
D	05-22 04:56:29.228	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	onProgressUpdate get param 100 : Tread is main
D	05-22 04:56:29.248	2232	cn.com.sgmsc.asynctask_PGD	SGMSC	onPostExecute : Tread is main
W	05-22 04:56:29.918	61	system_process	InputManagerService	Window already focused, ignoring focus gain of:
D	05-22 04:56:31.538	61	system_process	dalvikvm	GC_CONCURRENT freed 955K, 44% free 4676K/8327K,
D	05-22 04:57:23.708	61	system_process	SntpClient	request time failed: java.net.SocketException:
D	05-22 05:02:23.737	61	system_process	SntpClient	request time failed: java.net.SocketException:
I	05-22 05:02:46.688	123	com.android.systemui	dalvikvm	Jit: resizing JitTable from 512 to 1024

图 9-12 从日志信息中跟踪各部分代码的执行过程

9.4 Service 组件

Service 是 Android 系统提供的运行在后台的一种服务程序，例如：播放音乐，后台数据计算（如记录用户的地理信息位置的改变），发出 Notification，检测 SD 卡上文件的变化，等等。它的地位和 Activity 的级别差不多，只不过没有 Activity 使用的频率高。

9.4.1 Service 的生命周期

Service 有自己的生命周期，它的生命周期方法比 Activity 要少一些，只有 onCreate()、onStart()和 onDestroy()。onCreate()方法是当 Service 第一次被创建时，由系统调用。onStart(Intent intent, int startId)方法是当 startService()方法启动 Service 时，该方法被调用。从 Android 2.0 以后的版本开始使用 onStartCommand 替换了之前的 onStart，启动时会自动调用该 Service 的 onStartCommand()方法。onDestroy()方法是当 Service 不再使用时，由系统调用。

Service 没有可交互的用户界面，它不能自己启动，必须要通过某一个 Activity 或其他的 Context 对象来启动。启动一个 Service 有两种方式，一种是通过调用 startService()启动一个 Service，另一种是使用 bindService()方法来绑定一个存在的 Service。不同的启动方式对 Service 生命周期的影响是不一样的。

1. 通过 startService 启动

通过 startService 启动 Service，在启动时会经历 onCreate→onStart(或 onStartCommand)过程。注意：如果 Service 已经启动了，当再次启动 Service 时，不会再执行 onCreate()方法，而是直接执行 onStart(或 onStartCommand)方法。

在 Service 停止时直接进入 onDestroy 过程。如果调用者自己直接退出而没有调用 stopService，Service 会一直在后台运行，直到下次调用者再次启动起来，并调用 stopService 后才会停止。无论之前调用过几次 startService，只要调用一次 stopService，将结束该 Service。

2. 通过 bindService 启动

通过 bindService 启动 Service，在启动时只会运行 onCreate，这个时候将调用者和 Service 绑定在一起。

如果调用者退出了，Service 也就会调用 onUnbind→onDestroy，与调用者同时退出。这就是所谓绑定的含义。

使用这两种方式启动 Service，其生命周期可通过图 9-13 来描述。

3. Service 的进程优先级

Service 拥有较高的优先级。一般在下列几种情况下都不会被系统 killed 掉。

如果 Service 正在调用 onCreate、onStart 或 onDestory 方法，那么用于当前 Service 的进程则变为前台进程以避免被 killed。

如果 Service 已经被启动，拥有它的进程仅次于可见的进程，而比不可见的进程重要，这就

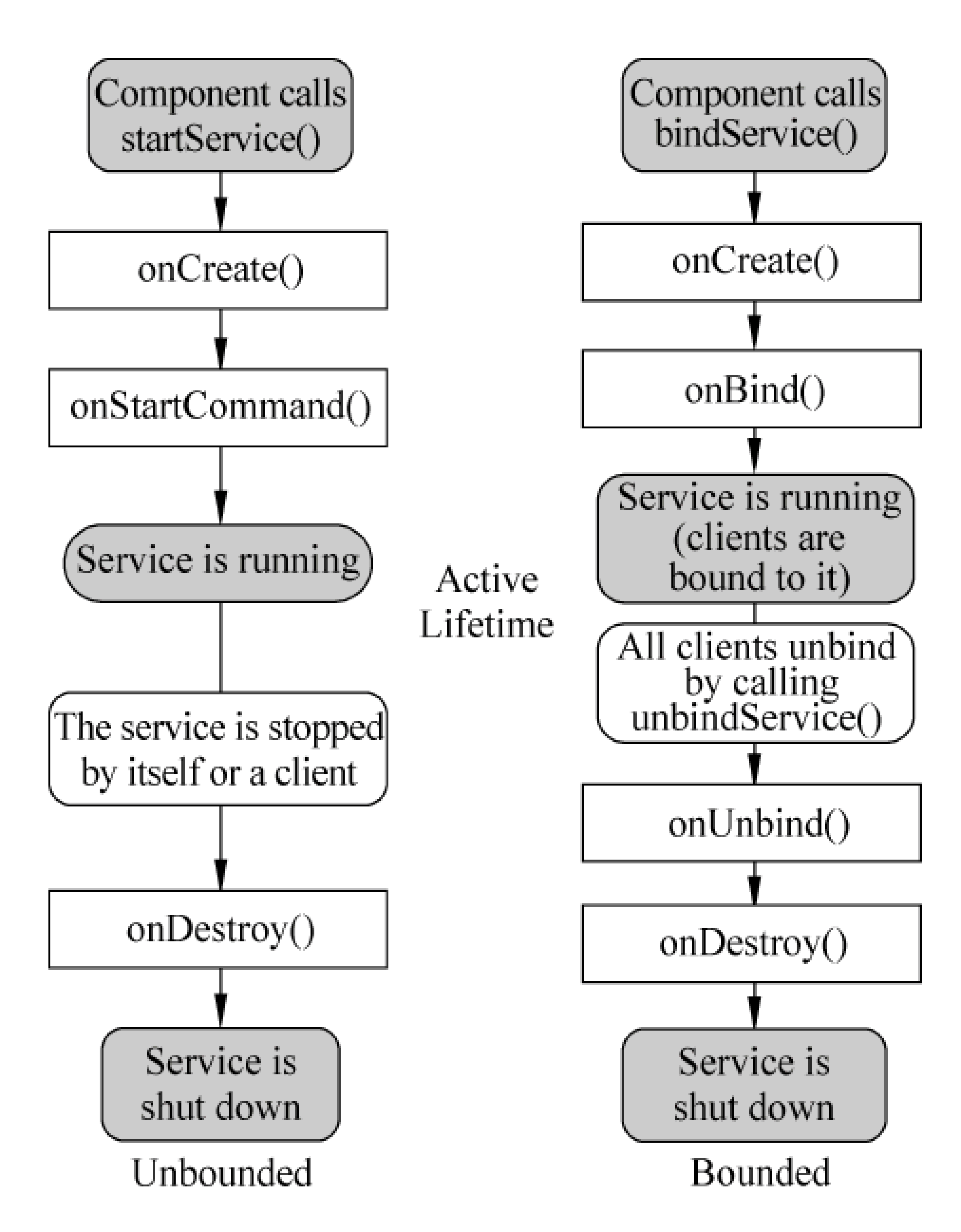


图 9-13　两种启动 Service 的生命周期表现

意味着 Service 一般不会被 killed。

如果客户端已经连接到 Service，那么拥有它的进程则拥有最高的优先级，可以认为该 Service 是可见的，不会被 killed。

如果 Service 可以使用 startForeground(int, Notification)方法来将 Service 设置为前台状态，那么系统就认为是对用户可见的，并不会在内存不足时将它 killed。

9.4.2　使用 Service

Service 位于 android.app 包下。Service 一般由 Activity 启动，但不依赖于 Activity。也可以由其他的 Service 或者 Broadcast Receiver 启动。

1. 创建 Service 子类需要重写的方法

使用 Service 进行编程，首先要创建一个 Service 类，创建方法比较简单，只要定义一个类继承于 Service，并且要重写该类中相应的方法即可。这些方法如下。

(1) onBind(Intent intent)方法。是必须实现的方法，返回一个绑定的接口给 Service。

(2) onCreate()方法。是当 Service 第一次被创建时，由系统调用。

(3) onStart(Intent intent, int startId)方法。是当 startService()方法启动 Service 时，该方法被调用。

(4) onDestroy()方法。是当 Service 不再使用时，由系统调用。

2. 启动和停止 Service

一旦定义好一个 Service，就可以在其他组件中启动该 Service 来使用它了。启动一个 Service 使用 Context.startService(Intent intent)方法，这与启动一个 Activity 非常相似，也是传递一个 Intent。或使用 bindService(Intent service, ServiceConnection conn, int flags)绑定

Service，其中第一个是 Intent；第二个是绑定 Service 的对象；第三个参数是创建 Service 的方式，一般指定为系统常量 BIND_AUTO_CREATE，即绑定时自动创建。

1）启动 Service

启动一个 Service 可通过类名称来显式启动，例如已经创建了一个名为 MyService 的类，则启动代码为：

```
Intent myIntent = new Intent(this, MyService.class);
myIntent.putExtra("TOPPING","Margherita");
startService(myIntent);
```

或使用隐式启动，代码为：

```
startService(new Intent(this, MyService.class));
```

2）停止 Service

停止一个 Service，可使用 stopSelf()方法，即由 Service 自己停止，也可使用 stopService()方法。例如代码：

```
stopService(new Intent(this, MyService.class));
```

3）注册 Service 组件

在应用程序中使用 Service，需要在 AndroiManifest. xml 文件中显式地注册＜service＞标签，例如：

```
<service android:enabled = "true" android:name = ".MyService"/>
```

3. Service 的应用

下面通过一个简单的例子来说明 Service 的应用。

【案例 9.5】 Service 与 BroadcastReceiver 联合应用：通过按钮启动 Service，Service 运行 1min 后停止，或通过单击“停止 Service”按钮中止 Service；通过 BroadcastReceiver 广播 Service 运行的时间。

【说明】 本例需要设计两个类，一个是 Activity 类，在其中布局按钮对象和文本框对象。另一个是继承 Service 的子类，重写其相关的方法，并且在该服务子类中定义 Thread 子类来创建后台线程。

案例中使用 BroadcastReceiver 来广播 Service 运行的时间。由于 BroadcastReceiver 接收的是应用程序的广播信息，所以使用动态注册 BroadcastReceiver 的方式。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 Service_BroadcastDemo 的 Android 项目。其应用程序名为 Service_Bdcast，包名为 cn. com. sgmsc. ServiceBroadcast，Activity 组件名为 Service_BdcastActivity。

(2) 准备字符串。编写 res/values 目录下的 strings. xml 文件，其代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>

    <string name = "app_name"> Service_BroadcastReceiver </string>
```

```
    <string name = "hello">请启动 Service</string>
    <string name = "myButton1">启动 Service</string>
    <string name = "myButton2">停止 Service</string>

</resources>
```

(3) 设计布局。编写 res/layout 目录下的 main.xml 文件。其代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"   android:layout_height = "fill_parent"
    android:gravity = "center_horizontal">
    <TextView
    android:id = "@ + id/myTextView"
        android:layout_width = "fill_parent"   android:layout_height = "wrap_content"
        android:background = " #55e1ef66"
        android:textSize = "20px"
        android:gravity = "center_horizontal"
        android:text = "@string/hello"/>

    <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"   android:layout_height = "fill_parent"
        android:gravity = "center_horizontal">
        <Button
            android:id = "@ + id/myButton1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "@string/myButton1"
            android:layout_margin = "10px"/>
        <Button
            android:id = "@ + id/myButton2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "@string/myButton2"
            android:layout_margin = "10px"/>
    </LinearLayout>
</LinearLayout>
```

(4) 开发逻辑代码。打开 src/cn.com.sgmsc.ServiceBroadcast 包下的 Service_BdcastActivity.java 文件,并编辑之。其代码如下所示。

```
package cn.com.sgmsc.ServiceBroadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.view.View.OnClickListener;
import android.view.Gravity;
import android.widget.Toast;
//……省略部分引入类

public class Service_BdcastActivity extends Activity {
 Button button1;                                     //声明按钮的引用
```

```
Button button2;                                        //声明按钮的引用
TextView myTextView;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myTextView = (TextView) this.findViewById(R.id.myTextView);
        button1 = (Button) this.findViewById(R.id.myButton1);
        button2 = (Button) this.findViewById(R.id.myButton2);

        //单击"启动 Service"按钮
        button1.setOnClickListener(new OnClickListener(){
          @Override
          public void onClick(View v) {
              Intent i = new Intent(Service_BdcastActivity.this, MyService.class);
              startService(i);
              Toast tst = Toast.makeText(Service_BdcastActivity.this,
                                  "Service 启动成功", Toast.LENGTH_SHORT);
              tst.setGravity(Gravity.CENTER, 0, 0);  //将提示信息显示在屏幕的中间位置
              tst.show();
          }
        });
        //单击"停止 Service"按钮
        button2.setOnClickListener(new OnClickListener(){
          @Override
          public void onClick(View v) {
              Intent i = new Intent(Service_BdcastActivity.this, MyService.class);
              stopService(i);
              Toast tst = Toast.makeText(Service_BdcastActivity.this,
                                  "Service 停止成功", Toast.LENGTH_SHORT);
              tst.setGravity(Gravity.CENTER, 0, 0);
              tst.show();
          }
        });

        //动态注册一个 BroadcastReceiver
        IntentFilter intentFilter = new IntentFilter("cn.com.sgmsc.ServBroad.myThread");
                                                       //创建过滤器
        MyBroadcasReceiver myBroadcasReceiver = new MyBroadcasReceiver();
        registerReceiver(myBroadcasReceiver, intentFilter); //注册 BroadcastReceiver 对象
    }

    //实现 MyBroadcasReceiver
    public class MyBroadcasReceiver extends BroadcastReceiver{
      @Override
      public void onReceive(Context context, Intent intent) {
        Bundle myBundle = intent.getExtras();
        int myInt = myBundle.getInt("myThread");
        if(myInt < 60){                                //后台 Service 运行 1min
            myTextView.setText("后台 Service 运行了" + myInt + "秒");
        }else{
            Intent i = new Intent(Service_BdcastActivity.this, MyService.class);
            stopService(i);
```

```
68                myTextView.setText("后台 Service 在" + myInt + "秒后停止运行");
69            }
70        }
71      }
72 }
```

① 第 19～55 行重写 onCreate()方法。

② 第 27～37 行，定义了第一个按钮的 onClickListener 的监听。第 30 行创建了一个 Intent 对象 i，第 31 行启动一个 Service，第 34 行设置 Toast 对象 tst 显示在屏幕的中间部位。注意，在这里使用了 Gravity，所以要在前面引入类 android. view. Gravity。

③ 第 52～54 行，动态注册了一个 BroadcastReceiver。第 52 行是创建一个 IntentFilter，其中"cn. com. sgmsc. ServBroad. myThread"是程序员自定义的，用于 Intent 的过滤匹配。

④ 第 58～71 行，定义一个广播接收器，重写接收广播后的响应事件 onReceive。第 61 行创建一个 Bundle 对象 myBundle，它获取传入的 Intent 中的附加信息，这个附加信息中有一个名为 myThread 的键-值对，其值记录着服务启动的时间。

⑤ 第 63～69 行，判断当服务启动的时间小于 60s，则对 TextView 中赋值并显示；当服务启动时间达到 60s，即终止服务，同时对 TextView 中赋值并显示。

（5）开发 MyService 类代码。在 src/cn. com. sgmsc. ServiceBroadcast 包下创建一个代码文件 MyService. java，该文件是定义一个服务类 MyService，它继承自 Service 类，并且重写其中的 onBind()、onStart()和 onDestroy()方法，并且定义一个 MyThread 子类。其代码如下所示。

```
package cn.com.sgmsc.ServiceBroadcast;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

public class MyService extends Service{
MyThread myThread;                                          //线程的引用
@Override
public IBinder onBind(Intent intent) {                      //重写的 onBind()方法
    return null;
}
@Override
public void onDestroy() {                                   //重写的 onDestroy()方法
    myThread.flag = false;                                  //停止线程运行
    super.onDestroy();
}
@Override
public void onStart(Intent intent, int startId) {           //重写 onStart()方法
    myThread = new MyThread() ;                             //初始化线程
    myThread.start();                                       //启动线程
    super.onStart(intent, startId);
}

//定义线程类
class MyThread extends Thread{
    boolean flag = true;                                    //循环标志位
    int c = 0;                                              //其值为发送的消息
    @Override
    public void run() {
        while(flag){
```

```
32              Intent i = new Intent("cn.com.sgmsc.ServBroad.myThread");//创建 Intent
33              i.putExtra("myThread", c);                    //放入数据
34              sendBroadcast(i);                             //发送广播
35              c++;
36              try{
37                  Thread.sleep(1000);                       //睡眠指定毫秒数
38              }catch(Exception e){                          //捕获异常
39                  e.printStackTrace();                      //打印异常
40              }
41          }
42      }
43  }
44 }
```

① 第 19～23 行重写 onStart()方法。在该方法中启动自定义的线程 MyThread。

② 第 26～43 行定义线程子类 MyThread。在其 run()方法中使用一个 while(true)循环，每隔 1 秒钟向 Intent 对象传入一个键-值对，其值为服务运行的时间；然后发送广播。

(6) 添加 Service 组件声明。在 AndroidManifest.xml 文件中声明了一个 Service 组件。其代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < manifest xmlns:android = "http://schemas.android.com/apk/res/android"
3      package = "cn.com.sgmsc.ServBroad"
4      android:versionCode = "1"
5      android:versionName = "1.0" >
6
7      < uses - sdk android:minSdkVersion = "10" />
8
9      < application
10         android:icon = "@drawable/ic_launcher"
11         android:label = "@string/app_name" >
12         < activity
13             android:label = "@string/app_name"
14             android:name = ".Service_Broad_Demo" >
15             < intent - filter >
16                 < action android:name = "android.intent.action.MAIN" />
17
18                 < category android:name = "android.intent.category.LAUNCHER" />
19             </intent - filter >
20         </activity>
21         < service
22           android:name = ".MyService">
23         </service >
24
25     </application >
26
27 </manifest >
```

第 21～23 行声明了一个 Service 组件，其定义在类 MyService 中。



图 9-14 初始运行界面

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Service_BroadcastDemo 项目。初始运行界面如图 9-14 所示，单击“启动 Service”按钮后，随即启动 MyService 服务，在 TextView 控件中显示 MyService 服务运行的时间，并且单击

按钮后在屏幕的中部出现启动服务的消息提示信息,如图 9-15 所示。然后,单击"停止 Service"按钮后,即停止 MyService 服务,并且在屏幕的中部出现停止服务的消息提示信息,如图 9-16 所示。如果不单击"停止 Service"按钮,在该服务启动 1min 后自动停止,此时在 TextView 控件中将显示"后台 Service 运行了 60 秒"。



图 9-15 单击"启动 Service"按钮后



图 9-16 单击"停止 Service"按钮后

9.5 Android 应用开发步骤及应用案例

9.5.1 应用开发的前期准备

开发 Android 应用项目,需求分析和概要设计一定要做到位,认真做好应用规划和架构设计。此阶段需要考虑的问题主要如下。

✍项目有哪些功能?
✍需要哪些用户界面?
✍各界面之间跳转的流程?
✍需要的数据及其数据的来源和格式?
✍是否需要服务端的支持?
✍是否需要本地数据库的支持?
✍是否需要特殊的权限?
✍是否需要后台进程? 等。

在程序员进入代码编写之前,都需要列出这个应用所必需的功能,该功能所需要的数据,数据的来源及存储;需要哪些展示界面,每个界面上需要显示哪些元素;并列出每个界面的跳转关系,需要哪些后台进程,涉及哪些权限,等等。对于比较复杂的界面,最好使用笔在纸上先画出草图,精心设计布局。

另一方面,一个应用系统都会有自己的一套图标集和色系基调,在编码前需要准备各界面显示的图标、图片、文字内容,以及音频与视频等素材,这些都是构成应用系统必不可少的资源。

9.5.2 应用开发步骤

1. 用户界面设计

按照项目规划文档,使用准备好的图标、图片等资源,在 XML 文件中描述出 Android 的用户界面。用户界面设计包括:界面布局模式设计,视图控件的大小、位置、色彩搭配等外观设计、控件中填充的数据适配器设计,菜单对话框设计。另外,还需要考虑其界面上各个控件需要响应的事件,如单击按钮触发事件,按键事件,触摸事件,单击 menu 选择菜单的响应事件,等等。

2. 数据操作和存储

在描述完应用程序的 UI 后,接下来需要仔细考虑数据的操作与存储策略。应用项目的数据来源可以是多方式的,如 SharedPreferences、文件、数据库、ContentProvider、网络的方式。在开发中,需要理清楚哪些数据需要存储,以及选择哪种方式存储数据。

3. 多页面的跳转实现

在应用的 UI 和数据存储都论证完成后,就可以把应用系统的整个流程连贯起来了,实现各个界面直接的调用和跳转流程。例如,是选择菜单实现跳转,还是单击按钮实现跳转,以及监听事件的处理。另外,需要考虑程序的健壮性,如果当一些跳转暂时无法连接时,可以使用其他的方式进行替代和弥补。

这个步骤用来进一步梳理应用的界面。如果整个流程都可以顺利地运行起来,那么应用项目的框架就已经完成,接下来就是进一步实现其后台处理与完善细节工作。

4. 增加 Service

前面做的工作都是比较"表面"的,主要实现的是人-机交互以及前台界面跳转等功能。在实际应用中还有一个比较重要的方面需要关注,那就是论证应用系统是否需要 Service,如果需要怎样实现自己的 Service。

5. 完善应用细节

细节决定成败。在完成大部分的功能后,最后还需要检查一些细节,例如,如果使用 Android 提供的某种特殊应用,在 AndroidManifest. xml 中是否添加了相关权限;如果对老版本进行了更新,那么在 AndroidManifest. xml 中是否更新 versionCode、versionName 的相应版本信息;在模拟器上显示的效果与真机显示的效果的细小区别,等等。这些细节技巧需要开发者自己在实践中不断积累,逐渐丰富自己的开发经验,夯实开发功底。

6. 应用测试

项目程序开发完成后,应该接收系统的应用测试。使用模拟器和 Eclipse 中的 DDMS 功能,可以完成大部分 Android 应用的测试工作,且测试出来的效果与真机上实现的效果非常接近。但是有一些操作,例如手机振动、打电话、拍照、传感和重力感应等功能还必须在真机上运行才能测试出来。所以对 Android 的应用测试最好是在模拟器与真机间结合进行。

7. 打包发布

在应用项目开发完成,并通过测试后,就可以发布了。发布应用需要对应用项目进行打

包、生成签名文件，这部分操作将在第 10 章中介绍。

9.5.3 音乐播放器案例

下面通过一个应用案例，来展示一个简单应用的开发步骤。

【案例 9.6】 设计一个单机版的音乐播放器。要求能对指定的音乐进行播放、暂停、停止控制；显示歌曲的相关信息；按下 menu 键时，弹出“退出”选项菜单，退出时需要有退出确认对话框。

【说明】 需要定义两个类，一个是 Activity 类，在其中布局按钮对象和显示音乐相关信息的若干控件对象。另一个是继承 Service 的子类，重写其相关的方法，并对音乐的播放、暂停、停止进行控制。

在两个类的 onCreate()方法中各注册一个广播接收器，并都需要定义各自的广播接收器子类，在 Activity 类中定义的广播接收器子类用于设置 start 按钮的显示图片、音乐播放状态变量 status 的值，并将用户的按键信息广播出去；在 Service 子类中定义的广播接收器子类用于设置控制音乐的播放、暂停、停止等操作，并将当前的播放状态广播出去。

【开发步骤及解析】

(1) 准备素材。本应用是播放一首歌曲，需要准备一个 mp3 的文件；需要有与歌曲相关的信息，如歌曲的词作与谱曲、歌词、演唱者；需要用于控制音乐播放的按钮图片、界面的背景图片等。

(2) 界面设计。因为本应用只是一个歌曲播放器，只需要一个用户界面。在这个界面中涉及的信息元素如下。

① 为配合歌曲内容，设置背景。

② 在界面的顶端放置按钮和歌曲主要信息。

✍左边并排放置两按钮，“播放”和“暂停”设置为一个按钮的两种状态。

✍右边纵向放置歌曲名称、歌唱人、作词作曲人信息。

③ 在界面下部以滚动的显示形式歌词内容。

④ 需要设计一个退出应用的选项菜单，并且选择退出时的确认对话框。

该音乐播放器运行时的几种效果如图 9-17～图 9-20 所示。



图 9-17 初始进入音乐播放器



图 9-18 正在播放歌曲




图 9-19 按下 menu 键显示退出菜单项



图 9-20 单击“退出”菜单项，弹出确认对话框

(3) 创建项目。在 Eclipse 中创建一个名为 MusicPlayer 的 Android 项目。其应用程序名为 MusicPlayer，包名为 cn. com. sgmsc. MusicPlayer，Activity 组件名为 MusicPlayerActivity。

(4) 准备资源。

① 将背景图片、按钮图片复制到 res\drawable-mdpi 下。

② 在 res 下创建子目录 raw，复制 htys. mp3 文件到其下。

③ 设置需要使用的颜色：colors. xml。

④ 设置歌曲歌词字符串数组：arrs. xml。

⑤ 设置歌曲信息字符串：strings. xml。

(5) 设计布局。编写 res/layout 目录下的 main. xml 文件。其代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:background = "@drawable/musicback"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
        <! -- 顶端按钮与歌曲名布局 -->
        < ImageButton
          android:id = "@ + id/start"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:src = "@drawable/png2"/>   <! -- "播放"按钮 -->
        < ImageButton
          android:id = "@ + id/stop"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:src = "@drawable/png1"/>   <! -- 播放"停止"按钮 -->
        < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
          android:orientation = "vertical"
```

```
24            android:layout_width="fill_parent"
25            android:layout_height="fill_parent">
26            <TextView
27                android:id="@+id/textView1"
28                android:layout_width="wrap_content"
29                android:layout_height="wrap_content"
30                android:textSize="20px"
31                android:textColor="#ffffff"
32                android:ellipsize="marquee"
33                android:layout_weight="1"
34                android:marqueeRepeatLimit="marquee_forever"
35                android:text="@string/myTextView1"/>
36            <TextView
37                android:id="@+id/textView2"
38                android:textSize="15px"
39                android:gravity="center_vertical"
40                android:layout_weight="1"
41                android:layout_width="wrap_content"
42                android:layout_height="wrap_content"
43                android:text="@string/myTextView2"/>
44            <TextView
45                android:id="@+id/textView3"
46                android:textSize="15px"
47                android:gravity="center_vertical"
48                android:layout_weight="1"
49                android:layout_width="wrap_content"
50                android:layout_height="wrap_content"
51                android:text="@string/myTextView3"/>
52        </LinearLayout>
53    </LinearLayout>
54    <ScrollView
55        android:layout_width="fill_parent"
56        android:layout_height="fill_parent"
57        android:fillViewport="true"
58        >
59        <!-- 歌词信息布局 -->
60        <ListView
61            android:id="@+id/singtext"
62            android:layout_width="fill_parent"
63            android:layout_height="fill_parent"
64            android:ellipsize="marquee"
65            />
66    </ScrollView>
67 </LinearLayout>
```

① 第 32 行，当显示的文字内容过长时，使用 android:ellipsize 属性设置显示方式。这里，android:ellipsize="marquee"是设置为走马灯方式，EditText 不支持 marquee 方式。

② 第 34 行，在 ellipsize 指定为 marquee 的情况下，android:marqueeRepeatLimit 设置重复滚动的次数，当设置为 marquee_forever 时表示无限次。

③ 第 54～66 行声明一个 ScrollView 控件，在其中定义一个 ListView 用于显示歌词。

(6) 开发逻辑代码。打开 src/cn.com.sgmsc.ServiceBroadcast 包下的 Service_BdcastActivity.java 文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.MusicPlayer;

```

```
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.BroadcastReceiver;
import android.content.DialogInterface;
import android.view.Gravity;
//…… 省略部分引入类

public class MusicPlayerActivity extends Activity implements OnClickListener{
ImageButton start;                                        //"播放/暂停"按钮
ImageButton stop;                                         //"停止"按钮
ActivityReceiver activityReceiver;
int status = 1;                   //当前的状态, 1 没有声音播放, 2 正在播放声音, 3 暂停

ListView lv;

    @Override
    public void onCreate(Bundle savedInstanceState) {     //重写的 onCreate()方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                    //设置当前的用户界面

        lv = (ListView)findViewById(R.id.singtext);       //获得 ListView 对象的引用
        BaseAdapter myAdapter = new BaseAdapter(){        //为 ListView 准备内容适配器
         //定义歌词字符串的数组
         String[] singtxts = getResources().getStringArray(R.array.singtexts);

         public int getCount() {return singtxts.length;}
                                        //歌词字符串的数组的总条目数,这里共 23 条
         public Object getItem(int arg0) { return null; }
         public long getItemId(int arg0) { return 0; }
         public View getView(int arg0, View arg1, ViewGroup arg2) {
             //动态生成每个下拉项对应的 View,每个下拉项 View 由 LinearLayout
             //中包含一个 TextView 构成
             LinearLayout ll = new LinearLayout(MusicPlayerActivity.this);
                                                          //初始化 LinearLayout
            ll.setOrientation(LinearLayout.HORIZONTAL);   //设置朝向
            ll.setPadding(3,0,0,0);                       //设置列表框的四周留白
            TextView tv = new TextView(MusicPlayerActivity.this);    //初始化 TextView
            tv.setText(singtxts[arg0].toString());        //设置内容
            tv.setTextSize(15);                           //设置字体大小
            tv.setTextColor(MusicPlayerActivity.this.getResources()
                        .getColor(R.color.stxt));         //设置字体颜色
            tv.setPadding(3,0,0,0);                       //设置文本控件的四周留白
            tv.setGravity(Gravity.LEFT);                  //设置文字居左对齐
            ll.addView(tv);                               //添加到 LinearLayout 中
            return ll;
         }
     };
        lv.setAdapter(myAdapter);
        lv.setVisibility(View.INVISIBLE);                 //设置 lv 不可见

        start = (ImageButton) this.findViewById(R.id.start); //得到 start 的引用
        stop = (ImageButton) this.findViewById(R.id.stop);   //得到 stop 按钮的引用
        start.setOnClickListener(this);                   //为按钮添加监听
        stop.setOnClickListener(this);                    //为按钮添加监听
        activityReceiver = new ActivityReceiver();        //创建 BroadcastReceiver
```

```
            IntentFilter filter = new IntentFilter();            //创建 IntentFilter 过滤器
            filter.addAction("cn.com.sgmsc.MusicPlayer.update"); //添加 Action
            registerReceiver(activityReceiver, filter);          //注册监听
            Intent intent = new Intent(this, MusicService.class); //创建 Intent
            startService(intent);                                //启动后台 Service
      }

  /* 自定义的 BroadcastReceiver */
  public class ActivityReceiver extends BroadcastReceiver{
      //…… 省略一: 省略了该方法的代码段,相应代码将在后面给出解析
  }

  /* 接口中的方法 */
  @Override
  public void onClick(View v) {
      //…… 省略二: 省略了该方法的代码段,相应代码将在后面给出解析
  }

  @Override
  protected void onDestroy() {                                   //释放时被调用
      super.onDestroy();
         Intent intent = new Intent(this, MusicService.class);   //创建 Intent
         stopService(intent);                                    //停止后台的 Service
  }

  /* 退出菜单及对话框 */
  @Override
  public boolean onCreateOptionsMenu(Menu menu){                 //弹出菜单
      menu.add(0,Menu.FIRST,0,"退出")
          .setIcon(android.R.drawable.ic_menu_delete);           //设置图标
      return true;
  }
  @Override
  public boolean onOptionsItemSelected(MenuItem item){           //选择的菜单项
      switch(item.getItemId()){                                  //分支判断
      case Menu.FIRST:
          showDialog(1);                                         //显示对话框
          break;
      }
      //将来可在此进行扩展
      return false;
  }
  @Override
  protected Dialog onCreateDialog(int id){                       //创建对话框
      switch(id){                                                //判断
      case 1:
          return new AlertDialog.Builder(this)
              .setTitle("您确定退出?")
              .setPositiveButton("确定", new android.content.DialogInterface
                                          .OnClickListener(){
                  @Override
                  public void onClick(DialogInterface dialog, int which) {
                      System.exit(0);                            //直接退出
                  }
              })
```

```
112             .setNegativeButton("取消", null)                //"取消"按钮
113             .create();
114     default:
115         return null;
116     }
117   }
118
119 }
```

① 第 19～62 行，重写 onCreate()方法。

② 第 24～48 行是创建一个 BaseAdapter 对象，并将歌词内容从字符串数组 singtxts 填充到 myAdapter 中，用于为 ListView 准备内容适配器。

③ 第 50 行，设置 lv 不可见。setVisibility()方法可以设置控件的可见与隐藏状态。

④ 第 56～59 行，onCreate()方法中动态注册广播接收器 activityReceiver，这个接收器是接收 action 为"cn. com. sgmsc. MusicPlayer. update"的 Intent 对象。

⑤ 第 60～61 行，启动一个服务，其服务定义在子类 MusicService 中。

⑥ 第 76～80 行，重写 onDestroy()方法，在该方法中停止服务。

⑦ 第 84～88 行，创建选项菜单。

⑧ 第 90～98 行，定义单击菜单项后的响应操作。这里是弹出普通类型的对话框。

⑨ 第 100～117 行，创建对话框，并定义单击相应按钮的响应操作。

接下来对省略的部分代码作解析。

省略一：这部分省略的代码是定义一个内部类 ActivityReceiver，该类继承自 BroadcastReceiver。该子类接收从服务子类中传入的 Intent 对象，因为对音乐的播放是在服务子类中进行的，这个 Intent 携带了音乐播放的当前状态。具体代码如下。

```
1    /* 自定义的 BroadcastReceiver */
2    public class ActivityReceiver extends BroadcastReceiver{
3      @Override
4      public void onReceive(Context context, Intent intent) {     //重写的 onReceive()方法
5          int mupdate = intent.getIntExtra("musicupdate", -1);//得到 intent 中的数据
6          switch(mupdate){                                          //分支判断
7          case 1:                                                   //没有声音播放
8              start.setImageResource(R.drawable.png2);              //更换图片，此时为播放图片
9              status = 1;                                           //设置当前状态
10             break;
11         case 2:                                                   //正在播放声音
12             start.setImageResource(R.drawable.png3);              //更换图片，此时为暂停图片
13             status = 2;                                           //设置当前状态
14             break;
15         case 3:                                                   //暂停中
16             start.setImageResource(R.drawable.png2);              //更换图片，此时为播放图片
17             status = 3;                                           //设置当前状态
18             break;
19         }
20     }
21   }
```

① 第 5 行，从广播的 Intent 中获取键-值对中键名为 musicupdate 的值，该值记录的是播放声音的当前状态。播放状态有三种：值为 1 时表示没有声音播放状态，值为 2 时表示正在播放声音状态，值为 3 时表示暂停状态。

② 第 6～19 行，根据当前的状态设置 start 按钮的图片和 status 变量的值。

省略二：这部分省略代码是本类接口 OnClickListener 中的回调方法。在该 onClick()方法中，创建一个 Intent 对象，并根据哪个按钮被单击发送不同的广播信息。

```
1    /* 接口中的方法 */
2    @Override
3    public void onClick(View v) {
4      Intent intent = new Intent("cn.com.sgmsc.MusicPlayer.control");        //创建 Intent
5      switch(v.getId()){                                      //分支判断
6      case R.id.start:                                        //单击“播放/暂停”按钮
7          lv.setVisibility(View.VISIBLE);
8          intent.putExtra("ACTION", 1);                       //存放数据
9          sendBroadcast(intent);                              //发送广播
10         break;
11     case R.id.stop:                                         //单击“停止”按钮
12         intent.putExtra("ACTION", 2);                       //存放数据
13         sendBroadcast(intent);                              //发送广播
14         break;
15     }
16   }
```

① 第 4 行创建一个 Intent 对象，该 Intent 的 action 内容是“cn.com.sgmsc.MusicPlayer.control”。

② 第 6～10 行定义，当 start 按钮被单击时，使用主界面的 lv 变为可见状态，给 ACTION 键-值对赋值 1，将该 Intent 广播出去。

③ 第 11～15 行定义，当 stop 按钮被单击时，给 ACTION 键-值对赋值 2，将该 Intent 广播出去。

(7) 开发 MusicService 类代码。在 src/cn.com.sgmsc.MusicPlayer 包下创建一个代码文件 MusicService.java，该文件是定义一个服务类 MusicService，它继承自 Service 类，并且重写其中的 onBind()、onCreate()和 onDestroy()方法，并且定义一个 ServiceReceiver 广播接收器子类。其代码如下所示。

```
package cn.com.sgmsc.MusicPlayer;

import android.app.Service;
import android.content.BroadcastReceiver;
import android.media.MediaPlayer;
import android.os.IBinder;
//…… 省略了部分引入类

public class MusicService extends Service{
MediaPlayer mp;

ServiceReceiver serviceReceiver;
int status = 1;                 //当前的状态, 1 没有声音播放, 2 正在播放声音, 3 暂停
@Override
public IBinder onBind(Intent intent) {                      //重写的 onBind()方法
    return null;
}
@Override
public void onCreate() {                                    //重写的 onCreate()方法
    status = 1;
    serviceReceiver = new ServiceReceiver();                //创建 BroadcastReceiver
```

```
22      IntentFilter filter = new IntentFilter();                    //创建过滤器
23      filter.addAction("cn.com.sgmsc.MusicPlayer.control");        //添加 Action
24      registerReceiver(serviceReceiver, filter);                   //注册 BroadcastReceiver
25      super.onCreate();
26  }
27  @Override
28  public void onDestroy() {                                        //重写的 onDestroy()方法
29      unregisterReceiver(serviceReceiver);                         //取消注册
30      super.onDestroy();
31  }
32
33  public class ServiceReceiver extends BroadcastReceiver{          //自定义 BroadcastReceiver
34      @Override
35      public void onReceive(Context context, Intent intent) {      //重写的响应方法
36          int action = intent.getIntExtra("ACTION", -1);
37          switch(action){
38          case 1:                                                  //播放或暂停声音
39              if(status == 1){                                     //当前没有声音播放
40                  mp = MediaPlayer.create(context, R.raw.htys);
41                  status = 2;
42                  Intent sendIntent = new Intent("cn.com.sgmsc.MusicPlayer.update");
43                  sendIntent.putExtra("musicupdate", 2);
44                  sendBroadcast(sendIntent);
45                  mp.start();
46              }
47              else if(status == 2){                                //正在播放声音
48                  mp.pause();                                      //停止
49                  status = 3;                                      //改变状态
50                  Intent sendIntent = new Intent("cn.com.sgmsc.MusicPlayer.update");
51                  sendIntent.putExtra("musicupdate", 3);           //存放数据
52                  sendBroadcast(sendIntent);                       //发送广播
53              }
54              else if(status == 3){                                //暂停中
55                  mp.start();                                      //播放声音
56                  status = 2;                                      //改变状态
57                  Intent sendIntent = new Intent("cn.com.sgmsc.MusicPlayer.update");
58                  sendIntent.putExtra("musicupdate", 2);           //存放数据
59                  sendBroadcast(sendIntent);                       //发送广播
60              }
61              break;
62          case 2:                                                  //停止声音
63              if(status == 2 || status == 3){                      //播放中或暂停中
64                  mp.stop();                                       //停止播放
65                  status = 1;                                      //改变状态
66                  Intent sendIntent = new Intent("cn.com.sgmsc.MusicPlayer.update");
67                  sendIntent.putExtra("musicupdate", 1);           //存放数据
68                  sendBroadcast(sendIntent);                       //发送广播
69              }
70          }
71      }
72   }
73  }
```

① 第 19～26 行重写 onCreate()方法。在该方法中设置初始的播放状态为无声音的；注册一个广播接收器 serviceReceiver，接收 action 为“cn.com.sgmsc.MusicPlayer.control”的 Intent 对象。

② 第 28～31 行重写 onDestroy()方法。在该方法中注销广播接收器 serviceReceiver。

③ 第33～72行定义了一个ServiceReceiver广播接收器子类。重写了onReceiver()方法对action为"cn.com.sgmsc.MusicPlayer.control"的Intent进入处理。

④ 第38～61行是当ACTION值为1,即start按钮被单击时的处理。此时要根据status中的状态值分别处理。第39～46行是定义当前处于没有播放任何音乐状态处理操作。其处理代码首先要实例化一个MediaPlayer对象mp,定义其音乐来源于raw目录下的htys.mp3文件;将status值设为2;创建一个Intent对象sendIntent,其action为"cn.com.sgmsc.MusicPlayer.update";向键-值对的musicupdate中传入值2,即将当前的音乐播放状态传入该键-值对中;最后将sendIntent广播出去。接下来的第47～53行,第54～60行,都是对start按钮的不同状态进行处理。

⑤ 第62～70行是当ACTION值为2,即stop按钮被单击时的处理代码。

(8) 添加Service组件声明。在AndroidManifest.xml文件中要声明一个Service组件。由于两个BroadcastReceiver都是在程序中动态注册的,所以不需要在AndroidManifest.xml文件中注册了。

【运行结果】 在Eclipse中启动Android模拟器,然后运行MusicPlayer项目。运行结果符合设计要求。

问一下:

为什么案例9.5在Service子类只重写了onStart()方法,而案例9.6在Service子类中只重写了onCreate()方法?

案例9.5和案例9.6都是关于Service的应用。

在案例9.5中,用户可以反复地对指定的Service进行启动和停止。因此在自定义的Service子类中,需要重写onStart()方法,以便每次启动该Service时,可以做些初始化的操作。

在案例9.6中,用户只是通过启动一次服务,就可对所播放的音乐进行反复的播放、暂停与停止操作,退出服务后程序便结束了。因此在自定义的Service子类中,需要重写onCreate()方法,只在Service子类首次创建时运行onCreate()方法,执行与创建该Service有关的操作。

小结

本章介绍了Android应用项目编程的几种后台处理技术。消息提示Toast和Notification给出的消息提示方式不同,在编程中要根据不同的应用场合选择使用。BroadcastReceiver传递的广播要与其注册时Intent的action内容相匹配。后台线程与UI线程的信息交互可使用Handler和AsyncTask两种方式,使用中要掌握它们的消息传递机制、正确的编程步骤,以及相关规则。Service是Android中的重要组件之一,在应用上其使用频率仅次于Activity。当Service提供的后台服务需要与用户进行交互时,这时通常借助于BroadcastReceiver来与前台的Activity交换信息。熟练掌握这些后台编程技术,就为深层次的应用开发奠定了不可或缺的基础。最后,简要地介绍了开发一个应用项目应该遵从的开发步骤。

随着无线互联网技术的发展,手机应用只有与网络相联,才能延展其生命力。第10章中将学习Android的网络应用开发。



练习

为自己的微博相册浏览模块添加一个可控制的背景音乐播放功能。



网络与位置地图

Google 公司以其强大的互联网搜索引擎业务,在互联网领域独占鳌头。Android 是由 Google 公司推出的一款手机操作系统,其网络功能自然会非常强大。随着无线互联网的迅猛发展,人们可以不受时间、空间的限制,随时随地地进行数据交换、网页浏览、事务处理,例如手机银行、手机炒股、手机地图、手机微博、手机 QQ 聊天、手机购物、手机收发邮件等。

本章主要介绍 Android 平台下进行网络编程和网络应用的相关知识,内容包括网络的通信方式,获取网络中数据,浏览网页以及地理位置定位等。

10.1 使用 Socket 进行通信

在 Android 平台下进行 Socket 开发和在 Java 平台下的开发比较类似。Java 在包 java.net 中提供了两个类 Socket 和 ServerSocket,分别用来表示双向连接的客户端和服务端。使用 Socket、ServerSocket 编程方式可以说是比较底层的,其他的高级协议(如 HTTP)都是建立在此基础之上的,而且 Socket 编程是跨平台的编程,可以在异构语言之间进行通信。所以掌握 Socket 网络编程是基础。

10.1.1 Socket 编程模型

Socket 通常称为"套接字",应用程序通常通过"套接字"向网络发送请求或者应答网络请求。使用 Socket 模型的运作机理是:首先创建一个 SocketServer 的类作为服务端,该服务端实现了多线程机制,可以在特定端口处监听多个客户请求,一旦有客户请求,Server 总是会创建一个服务来服务新来的客户,而自己继续监听。当客户端与服务器端建立连接后,在客户端和服务器端都会建立用于通信的 Socket 实例,接下来就是由各个 Socket 分别打开各自的输入、输出流,完成所需的会话。

客户端和服务器端通过套接字进行通信,主要的 Socket 类型有流套接字(Stream Socket)和数据报套接字(Datagram Socket),可以使用 TCP 或者 UDP,不同的底层协议有不同的 Socket。TCP 使用流套接字进行通信,UDP 使用数据报套接字进行通信。本节只介绍在 TCP/IP 协议族下的 Socket。

在 Socket 编程中,Socket 类用来建立客户端程序,ServerSocket 类用于建立服务器端程序。首先来介绍 ServerSocket 类编程。

1. ServerSocket 类

ServerSocket 类是实现一个服务器端的 Socket,利用这个类可以监听来自网络的请求。

创建 ServerSocket 的方法为：

```
ServerSocket(int localPort [,int queueLimit])
```

其中，参数 localPort 为端口号，其有效范围是 0～65 535。在创建时必须指定一个端口号，以便客户端能够向该端口号发送连接请求。参数 queueLimit 为可选项，用来显式地设置连接请求队列的长度，它将覆盖操作系统限定的队列的最大长度。

常用的操作 ServerSocket 的方法如下所述。

1）accept()方法

accept()方法为下一个传入的连接请求创建 Socket 实例，并将已成功连接的 Socket 实例返回给服务器套接字，如果没有连接请求，accept()方法将阻塞并等待。这里，“阻塞”是一个术语，它使程序运行暂时“停留”在这个地方，直到一个会话产生，然后程序继续。通常“阻塞”是由循环产生的。accept()是一种阻塞性方法，所谓阻塞性方法就是说该方法被调用后将等待客户的请求，直到有一个客户启动并请求连接到相同的端口，然后 accept()返回一个对应于客户的 Socket。

2）close()方法

close()方法用于关闭套接字。ServerSocket 的 close()方法使服务器释放占用的端口，并且断开与所有客户的连接。当一个服务器程序运行结束时，即使没有执行 ServerSocket 的 close()方法，操作系统也会释放这个服务器占用的端口。因此，服务器程序并不一定要在结束之前执行 close()方法。在某些情况下，如果希望及时释放服务器的端口，以便让其他程序能占用该端口，则可以显式地调用 ServerSocket 的 close()方法。

3）isClosed()方法

isClosed()方法用于判断 ServerSocket 是否关闭。只有执行了 ServerSocket 的 close()方法，isClosed()方法才返回 true；否则，即使 ServerSocket 还没有和特定端口绑定，isClosed()方法也会返回 false。

4）getInetAddress()方法

getInetAddress()方法返回一个 IP 地址，它使得 ServerSocket 获得服务器绑定的 IP 地址。

5）getLocalPort()方法

getLocalPort()方法返回一个 int 值，它使得 ServerSocket 获得服务器绑定的端口号。在构造 ServerSocket 时，如果把端口设为 0，那么将由操作系统为服务器分配一个端口（称为匿名端口），程序只要调用 getLocalPort()方法就能获知这个端口号。

2. Socket 类

使用 Socket 来与一个服务器通信，必须先在客户端创建一个 Socket，并在 Socket 对象中指定服务器的 IP 地址和端口，这也是使用 Socket 通信的第一步。创建 Socket 的方法为：

```
Socket(inetAddress remoteAddress ,int remotePort)
```

其中，参数 remoteAddress 是远程服务器的 IP 地址，这里，IP 地址可以由一个字符串来定义，这个字符串可以是数字型的地址（如 192.168.1.1），也可以是主机名（如 example.com）。参数 remotePort 是远程服务器的端口号。

利用 Socket 的构造函数，可以创建一个 TCP 套接字后，先连接到指定的远程地址和端口号上。由于使用 Socket 编程是一种网络通信应用，所以要在 AndroidManifest.xml 文件中的

<manifest>标签内添加网络访问权限,添加该权限的代码为:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

在编程中,操作 Socket 常用的方法如下。

(1) getInputStream()方法,功能是获得网络连接输入,同时返回一个 IutputStream 对象实例。

(2) getOutputStream()方法,功能是使得连接的另一端得到输入,同时返回一个 OutputStream 对象实例。

(3) close()方法,功能是关闭 Socket 对象。

接下来介绍 Socket 的通信过程,充分理解其通信过程,有助于 Socket 开发。

3. Socket 通信过程

Socket 通信过程主要分为服务器端编程和客户端编程。

服务器端,首先使用 ServerSocket 监听指定的端口,等待客户连接请求,客户连接后,会话产生;在完成会话后,关闭连接。

客户端,使用 Socket 对网络上某一个服务器的某一个端口发出连接请求,一旦连接成功,打开会话;会话完成后,关闭 Socket。

如果是多客户同时连接服务器,服务器中主程序监听一端口,等待客户接入;同时构造一个线程类,准备接管会话。当一个 Socket 会话产生后,将这个会话交给线程处理,然后主程序继续监听。

在 Android Socket 的通信中,中间的管道连接是通过 InputStream/OutputStream 流实现的,一旦管道建立起来就可以进行通信,如果对同一个 Socket 创建重复管道会异常。关闭管道的同时意味着关闭 Socket。在开发 Socket 程序时一定要注意通信的顺序。服务器端首先得到输入流,然后将输入流信息输出到其各个客户端;客户端是首先建立连接,然后写入输出流,最后再获得输入流。如果开发的顺序不对则会产生 EOFException 的异常。为此,给出使用 Socket 的开发步骤。

根据 Socket 的通信过程,得出其编程步骤。下面给出阐述。

4. Socket 开发步骤

1) 服务器端编程步骤

(1) 创建服务器端套接字并绑定到一个端口上(因为 0~1023 是系统预留的号码,端口号最好选用大于 1024 的数字)。

(2) 套接字设置监听模式等待连接请求。

(3) 接受连接请求后进行通信。

(4) 返回,等待下一个连接请求。

2) 客户端编程步骤

(1) 创建客户端套接字(指定服务器端 IP 地址与端口号)。

(2) 连接(Android 创建 Socket 时会自动连接)。

(3) 与服务器端进行通信。

(4) 关闭套接字。

最后别忘了，要在 AndroidManifest.xml 文件中的<manifest>标签内添加可访问网络权限，代码为"<uses-permission android:name="android.permission.INTERNET"/>"。

10.1.2　使用 Socket 应用实例

【案例 10.1】 使用 Socket 进行服务器与客户端之间的通信：当客户连接服务器成功后服务器向控制台传送一字符串信息；服务器端向客户端传送系统当前日期。

【说明】 本例需要开发服务器端程序和客户端程序。服务器端的程序是 Java Application，客户端程序是 Android Application。

本例的服务器端和客户端都创建在本机上，因此服务器的 IP 地址是本机的 IP 地址。在开发前应该先知道本机的 IP 地址。获取本机的 IP 地址可使用如下方法：在命令行状态，输入"ipconfig"，即可看到本机的 IP 地址信息，如图 10-1 所示。

```
C:\Windows\system32\cmd.exe

   媒体状态  . . . . . . . . . . . . : 媒体已断开
   连接特定的 DNS 后缀 . . . . . . . :

无线局域网适配器 无线网络连接:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::a55d:4a53:bed5:839d%11
   IPv4 地址 . . . . . . . . . . . . : 192.168.1.102
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.1.1

隧道适配器 isatap.{54C76B36-4772-4329-98AB-8A186A54A201}:

   媒体状态  . . . . . . . . . . . . : 媒体已断开
   连接特定的 DNS 后缀 . . . . . . . :

隧道适配器 Teredo Tunneling Pseudo-Interface:

   连接特定的 DNS 后缀 . . . . . . . :
   IPv6 地址 . . . . . . . . . . . . : 2001:0:4137:9e76:284d:2f9a:3f57:fe99
   本地链接 IPv6 地址. . . . . . . . : fe80::284d:2f9a:3f57:fe99%13
   默认网关. . . . . . . . . . . . . : ::

C:\Users\Linda>
```

图 10-1　在命令行状态下查看本机的 IP 地址

注意，如果计算机网络连接属性中设置本地连接是"自动获得 IP 地址"方式，则本机的 IP 地址是不固定的，有时会有变化。所以在开发前最好查看一下本机的 IP 地址。有时候，测试本案例程序时不能连接到服务器，有可能是 IP 地址不对引起的，可考虑检查程序中指定的 IP 地址是否与本机的 IP 地址一致。

【开发步骤及解析】

(1) 开发服务器端程序。

① 创建 Java Application 项目。在 Eclipse 中，选择菜单 New→Java Project，创建一个名为 Socket_Server 的 Java 应用项目。

② 创建包名。展开至 Socket_Server 的 src 目录，选择菜单 New→Package，输入包名为 cn.com.sgmsc.Socket。

③ 创建 Java 代码。展开至 Socket_Server/src 目录的包 cn.com.sgmsc.Socket，选择菜单 New→File，输入文件名为 MyServer.java，并编辑之，其代码如下所示。

```
package cn.com.sgmsc.Socket;

import java.io.DataOutputStream;       //引入相关类
import java.net.ServerSocket;          //引入相关类
import java.net.Socket;                //引入相关类
```

```
6  import java.util.Date;                //引入相关类
7
8  public class MyServer{
9   public static void main(String args[]){
10        try{
11            ServerSocket myss = new ServerSocket(8000);
12            System.out.println("Listening...");
13            while(true){
14            Socket ssocket = myss.accept();
15            System.out.println("Client Connected...");
16            DataOutputStream dout = new DataOutputStream(ssocket.getOutputStream());
17            Date dt = new Date();
18            dout.writeUTF(dt.toLocalString());
19            dout.close();
20            ssocket.close();
21        }
22     }
23     catch(Exception e){
24            e.printStackTrace();
25     }
26  }
27 }
```

✍第 11 行创建服务器端套接字 myss，并绑定到一个端口 8000 上。这时，服务器 Socket 就可以监听来自网络端口 8000 的请求。

✍第 12 行，使用 System.out.println()方法向系统的控制台发出字符串"Listening..."，表示服务器已启动，正在监听客户端。

✍第 13～21 行是一个无限循环，在这个循环中定义了一系列的服务器接到客户请求的处理，直到服务器程序退出。

✍第 14 行通过服务器 Socket 的 accept()方法创建一个 Socket 实例，如果此时有客户端已成功连接到则创建 Socket 实例完成，如果没有客户端的连接请求，accept()方法将阻塞并等待。当等待到客户端的连接请求后则创建 Socket 实例 ssocket，并向控制台发出客户端已连接信息（第 15 行定义）。

✍第 16 行创建一个 ssocket 的输出流对象 dout，该对象用于向客户端传送信息；第 18 行，利用 writeUTF()方法将系统当前日期的字符串值写入到 dout 中。

✍第 19、20 行，当向客户端传出信息后即刻释放输出流 dout 和 Socket 对象 ssocket。然后返回到循环头等待下一个连接请求。

（2）开发客户端编程。

① 创建项目。在 Eclipse 中创建一个名为 Socket_Client 的 Android 项目。其应用程序名为 Socket_Client，包名为 cn.com.sgmsc.Socket，Activity 组件名为 ClientActivity。

② 准备字符串资源。编写 res/values 目录下的 strings.xml 文件。

③ 设计布局。编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
<EditText
```

```
android:id = "@ + id/et"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/et"
    android:editable = "false"
    android:cursorVisible = "false"
    />
</LinearLayout>
```

④ 开发逻辑代码。打开 src/cn. com. sgmsc. Socket 包下的 ClientActivity. java 文件，并编辑之。代码如下所示。

```
1  package cn.com.sgmsc.Socket;
2
3  import java.io.DataInputStream;
4  import java.net.Socket;
5  import android.app.Activity;
6  import android.os.Bundle;
7  import android.widget.EditText;
8  //import java.net.InetAddress;
9
10 public class ClientActivity extends Activity {
11   @Override
12   public void onCreate(Bundle savedInstanceState) {
13     super.onCreate(savedInstanceState);
14     setContentView(R.layout.main);                    //设置当前屏幕
15     connectToServer();                                //连接服务端
16   }
17
18   public void connectToServer(){                      //方法: 连接服务器端代码
19     try{
20       EditText myet = (EditText)findViewById(R.id.et);            //获得 EditText 对象
21       Socket mysocket = new Socket("192.168.1.102", 8000);       //创建 Socket 对象
22 //      InetAddress serverAddr = InetAddress.getByName("192.168.1.102");
                                                         //获取本机的 IP 地址
23 //      Socket mysocket = new Socket(serverAddr, 8000);            //创建 Socket 对象
24       DataInputStream dinstream = new DataInputStream(mysocket.getInputStream());
                                                         //获得 DataInputStream 对象
25       String servermsg = dinstream.readUTF();          //读取服务端发来的消息
26       myet.setText(servermsg);                         //设置 EditText 对象
27       mysocket.close();                                //关闭 Socket 对象
28     }
29     catch(Exception e){                                //捕获并打印异常
30       e.printStackTrace();
31     }
32   }
33 }
```

✍在 onCreate()方法内需要进行连接服务器端的编程，为了突出连接步骤，特将相关代码写在一个自定义的 connectToServer()方法中。

✍第 21 行创建客户端套接字 mysocket，并指定服务器端 IP 地址与端口号。实现这一步骤也可先指定 IP，再创建客户端套接字，见被注释的第 22、23 行，如果使用这种创建 Socket 的方法，则在前面一定要引入 java. net. InetAddress 类。mysocket 对象一旦创建，即刻与服务器连接上了。

✍第 24 行获得一个 DataInputStream 流对象 dinstream，用于从服务器中得到一个流信息。

✍第 25 行，通过 readUTF()方法从服务器的输出流对象中读入流信息。

✍第 26 行将从服务器中得到的相关信息显示在 EditText 中。

✍第 27 行套接字 mysocket。

⑤ 添加权限。打开根目录下的 AndroidManifest.xml，添加网络访问权限，其代码如下所示。

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3         package="cn.com.sgmsc.Socket"
4         android:versionCode="1"
5         android:versionName="1.0">
6       <uses-sdk android:minSdkVersion="10" />
7 
8       <application android:icon="@drawable/icon" android:label="@string/app_name">
9           <activity android:name=".ClientActivity"
10                    android:label="@string/app_name">
11              <intent-filter>
12                  <action android:name="android.intent.action.MAIN" />
13                  <category android:name="android.intent.category.LAUNCHER" />
14              </intent-filter>
15          </activity>
16 
17      </application>
18 
19        <uses-permission android:name="android.permission.INTERNET"/>
20 
21 </manifest>
```

第 19 行添加网络权限。

【运行结果】 在 Eclipse 中，首先启动服务器端的程序，方法是选择 Socket_Server 项目名称，然后选择菜单 Run→Run As→Java Application，即可在 Eclipse 工作窗口下方的 Console 窗口中看到如“Listening...”的信息，如图 10-2 所示。这表明服务器端程序已成功启动。

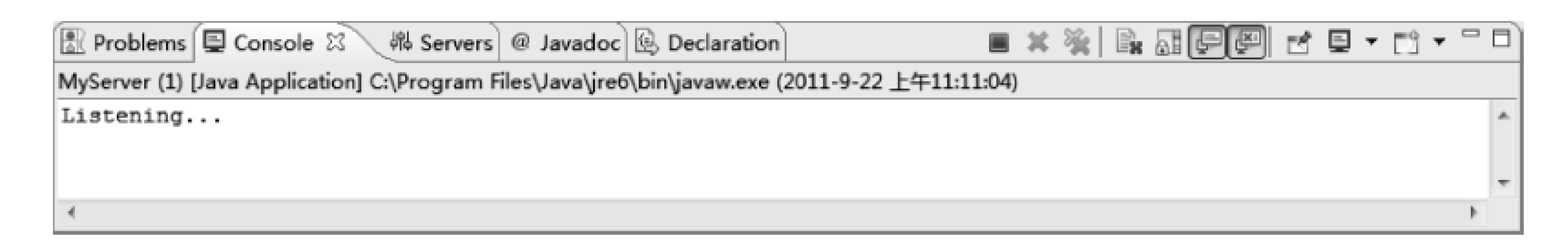


图 10-2　刚启动服务器端程序时的 Console 窗口

然后启动客户端程序。运行项目 Socket_Client。当连接上服务器后出现如图 10-3 所示的界面，这个日期时间值串来自于服务器中。

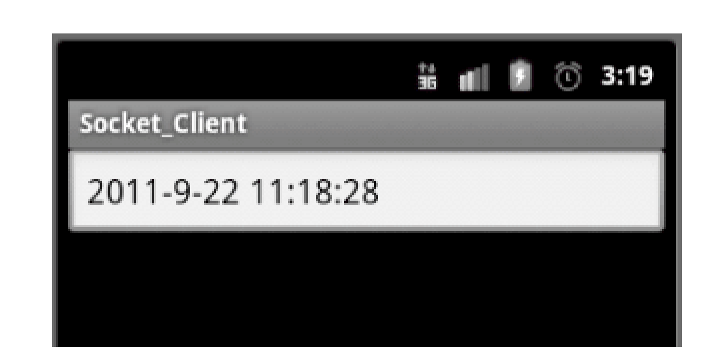


图 10-3　启动客户端程序的运行结果

这时再看一下 Eclipse 的 Console 窗口，会发现多些信息，如图 10-4 所示。

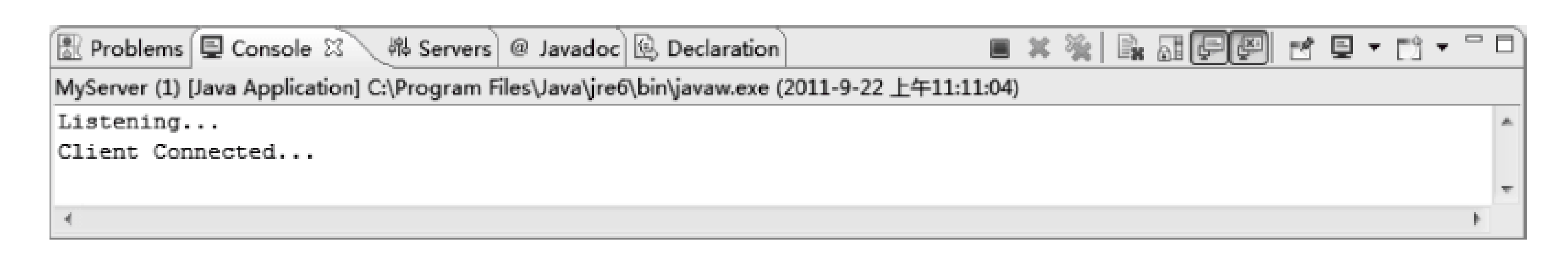


图 10-4　启动了客户端程序后的 Console 窗口

10.2 获取网络数据资源

由于移动设备的存储功能有限，有许多数据是保存在网络服务器中。本节介绍从网络上获取数据资源的应用。Android 获取网络上的资源有两种方式——使用 URL 和 HTTP。获取网络资源需要使用 Tomcat 服务器。下面首先介绍如何设置 Tomcat 服务器。

10.2.1 Eclipse 下的 Tomcat 设置

并不是所有的 Eclipse 版本都能安装 Tomcat 服务器。本书中使用的是 Eclipse 的 Jee 版，即 Eclipse IDE for Java EE Developers，可从网址 http://www.eclipse.org/downloads/中下载得到。Jee 版的 Eclipse，没有像 MyEclipse 那样集成了 Tomcat，需要自己设置。在 Eclipse 的 Jee 版中安装并设置 Tomcat 服务器的步骤如下。

第一步，下载并解压 Tomcat。下载 Tomcat 7.0.11，得到压缩包文件 apache-tomcat-7.0.11-windows-x86.zip，并解压。在本书中解压路径为 E:\ apache-tomcat-7.0.11。

第二步，在 Eclipse 中创建 Tomcat 服务器。选择菜单 File→New→Other…，打开 New 对话框，选择列表框中的 Server 下的 Server 项，如图 10-5 所示。

单击 Next 按钮，进入选择 Tomcat 版本的对话框。选择 Apache 下的 Tomcat v7.0 Server 项，如图 10-6 所示。注意：如果 Next 或 Finish 按钮都是灰的，即不可用状态，那么需要选择菜单 Window→Preferences→Server→Runtime Environments 进行设置，新建一个 Tomcat 运行环境，指定 Tomcat 的路径和 jre。

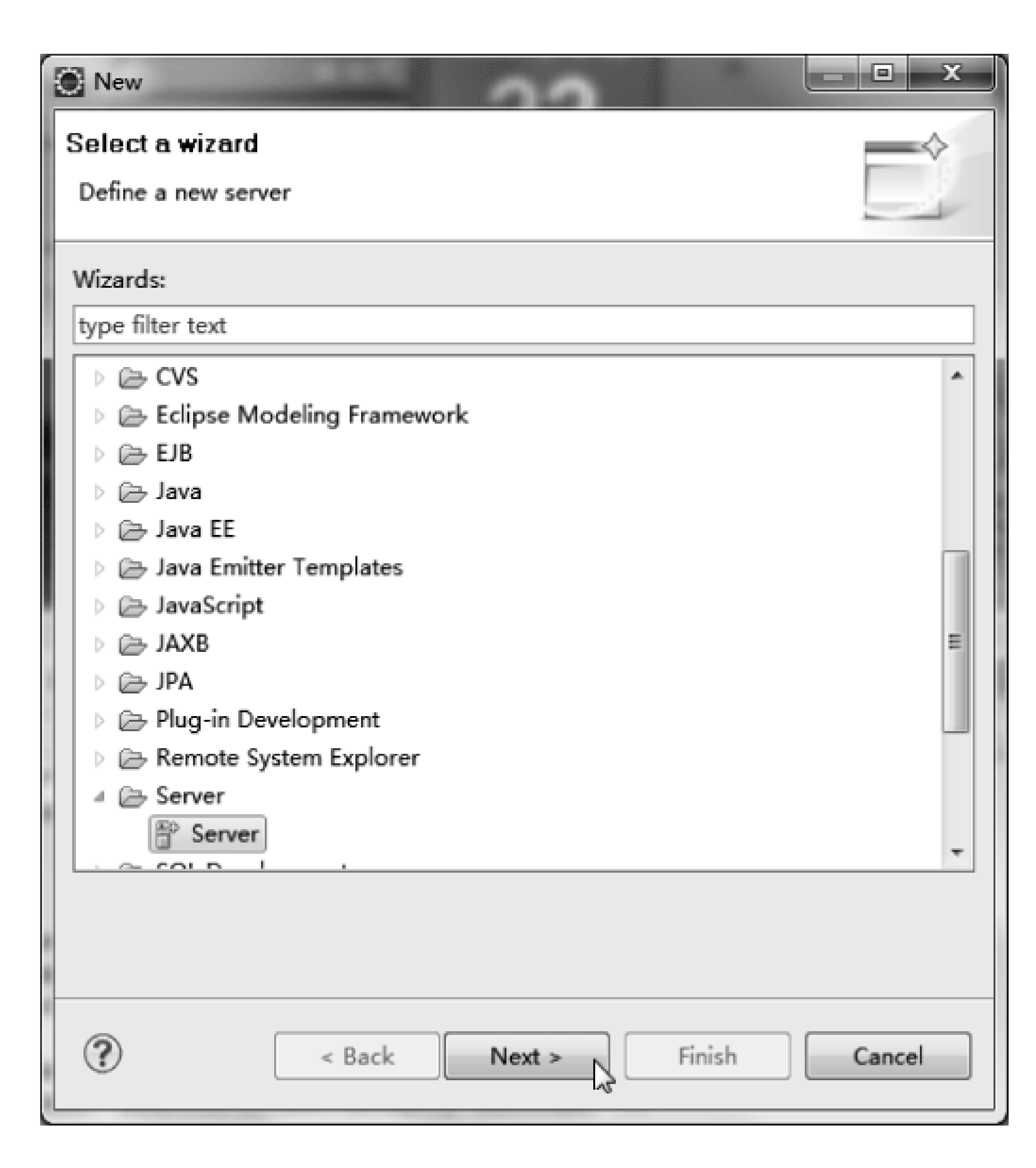




图 10-5 新建 Server

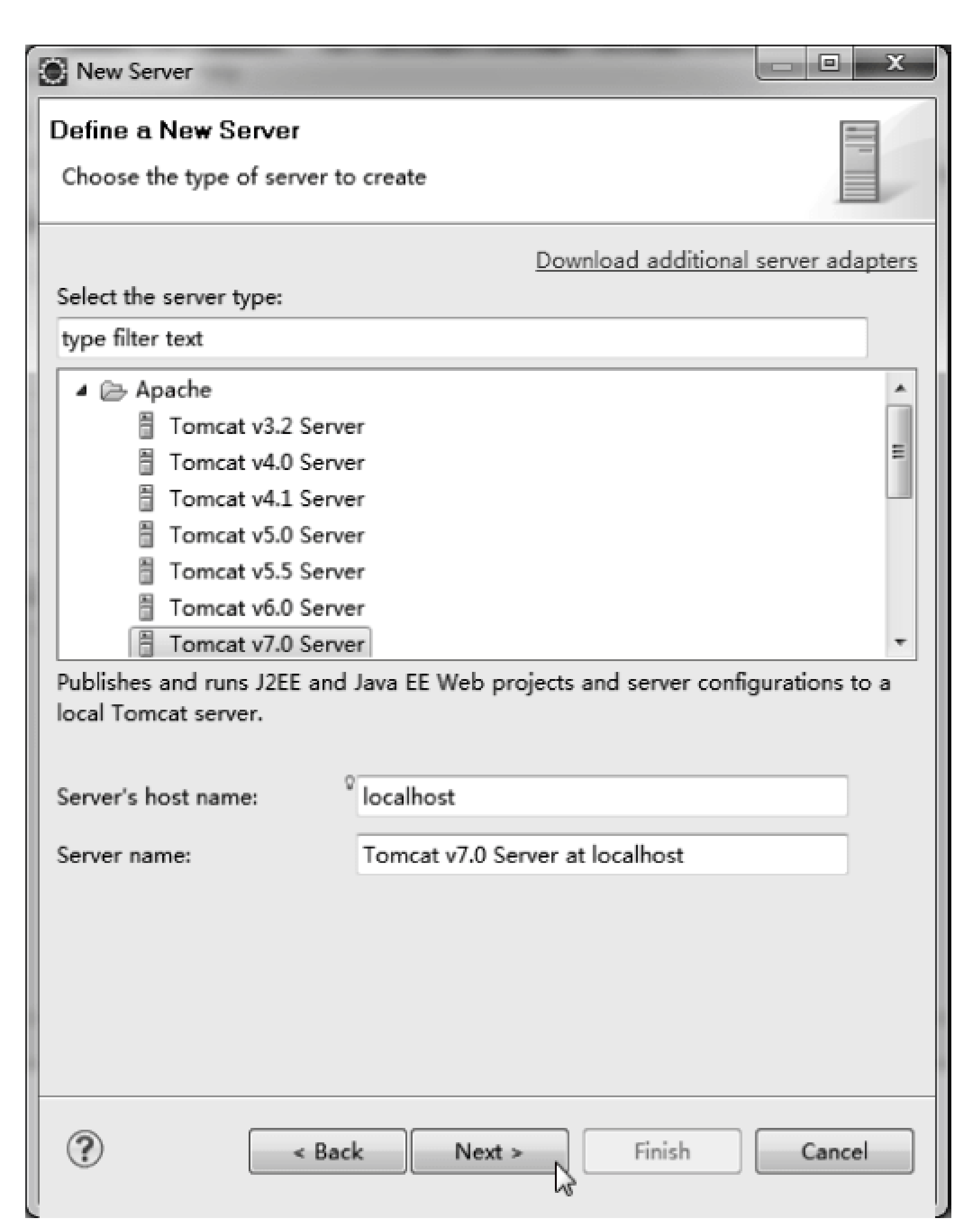


图 10-6 选择 Tomcat 版本

单击 Next 按钮，进入指定 Tomcat 安装文件夹的对话框。在 Tomcat installation directory 中输入 Tomcat 的安装文件夹（或单击 Browse…按钮，选择 Tomcat 的安装文件夹），

在 JRE 下拉列表中选择 jre7 项，单击 Finish 按钮，如图 10-7 所示。

第三步，在 Eclipse 中配置 Tomcat。进入 Eclipse 的 DDMS 页中，选择菜单 Window→Show View→Other...，如图 10-8 所示。进入 Show View 对话框，选择列表框中的 Server 下的 Server 项，如图 10-9 所示。



图 10-7 指定 Tomcat 安装文件夹

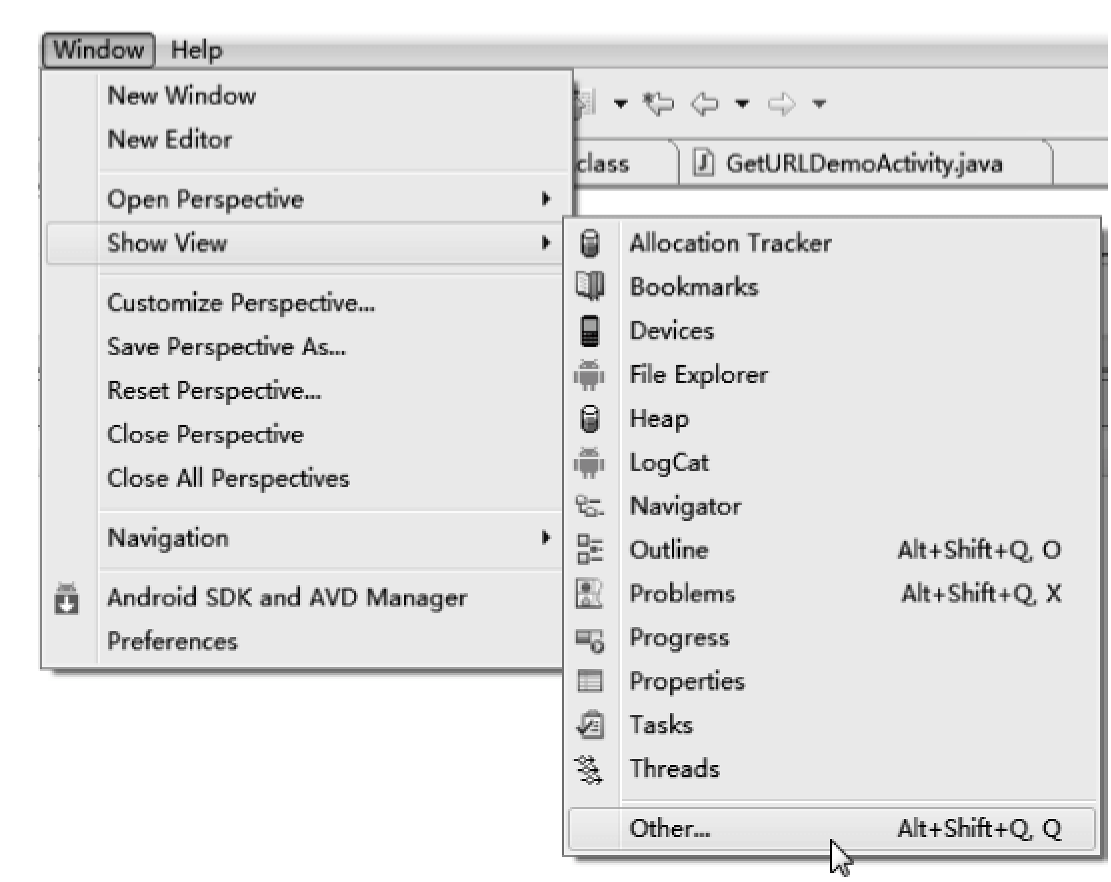


图 10-8 选择菜单过程

单击 OK 按钮，进入 DDMS 页的 Servers 窗口，如图 10-10 所示。

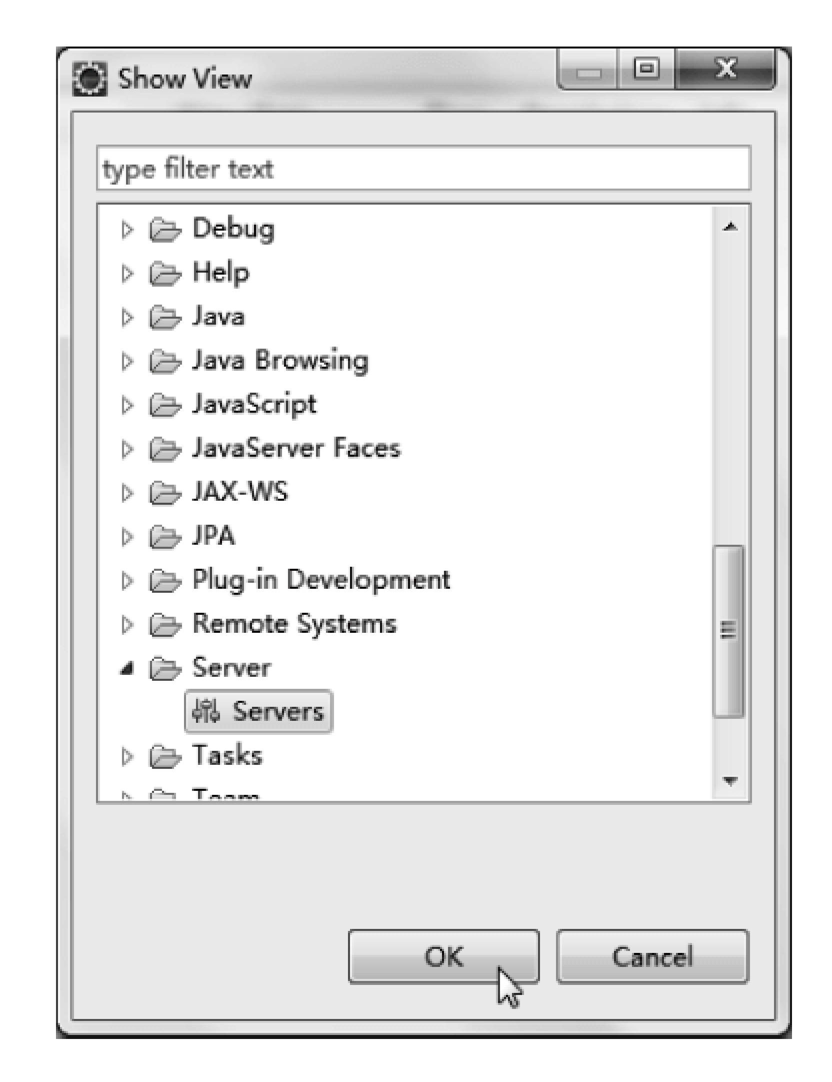


图 10-9 Show View 对话框

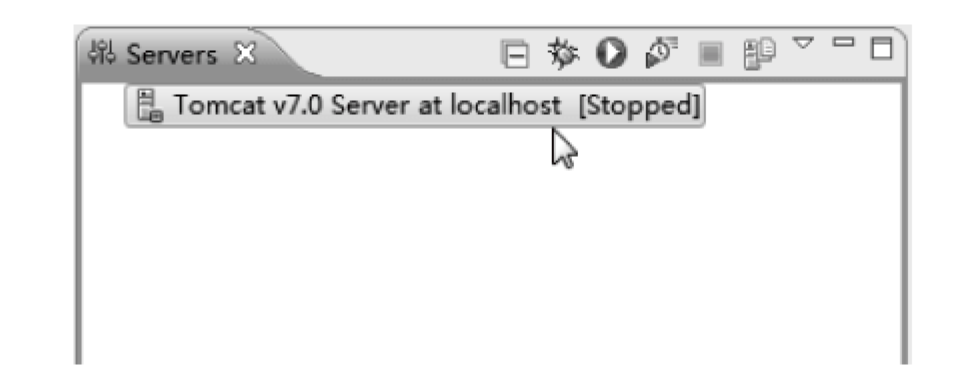


图 10-10 DDMS 页中的 Servers 窗口

双击 Tomcat v7.0 Server at localhost [Stopped]项，打开 Tomcat v7.0 Server at localhost 窗口，进行相关属性设置。在此窗口中主要对下列属性进行设置：在 Server Locations 里，选择 Use Tomcat installation(takes control of Tomcat installation)；在 Server Options 里，选择 Publish module contexts to separate XML files；如果项目比较大，可能启动时间较长，需要修

改 Timeouts 值；还可以设置 Tomcat 服务器的端口参数等，如图 10-11 所示。

图 10-11　Tomcat 服务器的属性设置

第四步，启动 Tomcat 服务器。在 Eclipse 的 DDMS 页的 Servers 窗口（如图 10-10 所示）或在 Eclipse 的 Java 页下部的 Servers 窗口（如图 10-12 所示）中的工具按钮中有一个"开启"按钮，单击该按钮即可启动 Tomcat 服务器。当 Tomcat 服务器是启动状态时，该工具按钮的"关闭服务"按钮可用。如果要关闭服务器，可单击该按钮。

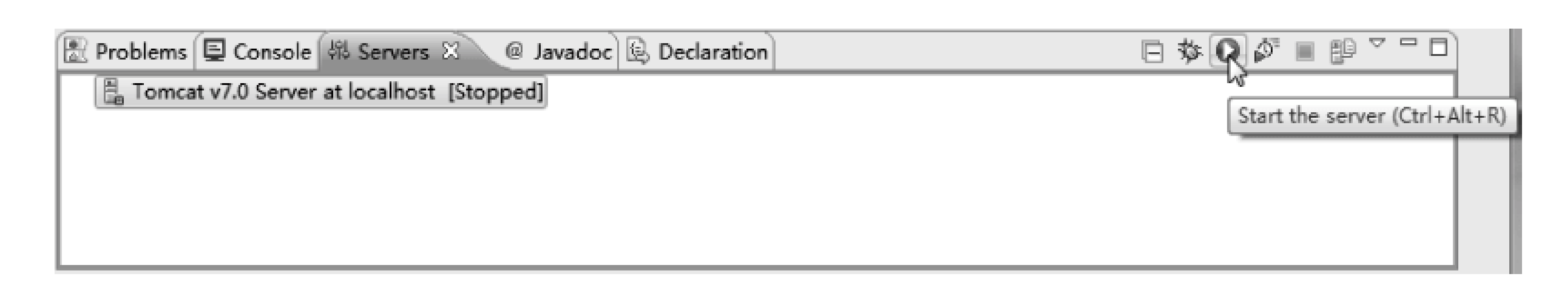


图 10-12　Java 页中的 Servers 窗口

至此，在 Eclipse Jee 中配置 Tomcat 服务器已经完成，并可启动 Tomcat 服务器。

10.2.2　通过 URL 获取网络资源

使用 URL 获取网络资源，需要使用 URL 类和 URLConnection 类，这两个类都位于 java.net 包下。下面通过一个案例来说明如何通过 URL 获取网络的资源。

【案例 10.2】　通过 URL 获取服务器中的一个 txt 文本文件和一个 png 图片文件，并显示在手机屏幕上。

【说明】　通过 URL 访问 Tomcat 服务器上的文件，这些文件一般都是放在 Tomcat 的 webapps 文件夹中的。

本例中，Tomcat 服务器安装在本地机上，本机的 IP 地址为 192.168.1.102，Tomcat 服务器的端口号为 8080。注意，编程时不要使用 localhost 或 127.0.0.1，因为这是 Android 模拟器的 IP 地址。

【开发步骤及解析】

（1）准备文件。在 Tomcat 的 webapps 文件夹中创建文件夹 MyURLFiles，分别将一个

txt 文本文件和一个 png 图片文件复制到新建的文件夹 MyURLFiles 中。

(2) 创建项目。在 Eclipse 中创建一个名为 GetURLMsg 的 Android 项目。其应用程序名为 GetURLDemo,包名为 cn.com.sgmsc.geturl,Activity 组件名为 GetURLDemoActivity。

(3) 设计布局。编写 res/layout 目录下的 main.xml 文件,代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <EditText
        android:id="@+id/et"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:editable="false"
        android:cursorVisible="false"
        android:layout_gravity="center_horizontal"
        />                  <!-- 声明一个 EditText 控件 -->
     <ImageView
        android:id="@+id/iv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        />                  <!-- 声明一个 ImageView 控件 -->
     <Button
        android:id="@+id/btn"
        android:text="@string/btn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        />                  <!-- 声明一个 Button 控件 -->
</LinearLayout>
```

(4) 开发逻辑代码。打开 src/cn.com.sgmsc.geturl 包下的 GetURLDemoActivity.java 文件,并编辑之,代码如下所示。

```
package cn.com.sgmsc.geturl;

import java.io.BufferedInputStream;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

import org.apache.http.util.ByteArrayBuffer;
import org.apache.http.util.EncodingUtils;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
import android. widget. ImageView;

public class GetURLDemoActivity extends Activity {
  String stringURL = "http://192.168.1.102:8080/MyUrlFiles/sgms_msg1.txt";
                                                          //指定服务器上的文本文件名
  String bitmapURL = "http://192.168.1.102:8080/MyUrlFiles/sgms_android.png";
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super. onCreate(savedInstanceState);
      setContentView(R. layout. main);
      Button btn = (Button)findViewById(R. id. btn);       //获得 Button 控件对象
      btn. setOnClickListener(new View. OnClickListener() {
          @Override
          public void onClick(View v) {                    //重写 onClick()方法
              getStringURLResources();                     //获得字符串资源
              getBitmapURLResources();                     //获得图片资源
          }
      });
  }
  //方法,根据指定 URL 字符串获取网络文本资源
  public void getStringURLResources(){
     try{
          URL myUrl = new URL(stringURL);                  //创建 URL 对象
          URLConnection myConn = myUrl. openConnection();//打开连接
          InputStream in = myConn. getInputStream();       //获取输入流
          BufferedInputStream bis = new BufferedInputStream(in);
                                                           //获取 BufferedInputStream 对象
          ByteArrayBuffer baf = new ByteArrayBuffer(bis. available());
          int data = 0;
          while((data = bis. read())!= -1){                //读取 BufferedInputStream 中数据
              baf. append((byte)data);                     //将数据读取到 ByteArrayBuffer 中
          }
          String msg = EncodingUtils. getString(baf. toByteArray(), "UTF-8");
                                                           //转换为字符串
          EditText et = (EditText)findViewById(R. id. et);//获得 EditText 对象
          et. setText(msg);                                //设置 EditText 控件中的内容
      }
      catch(Exception e){
          e. printStackTrace();
      }
  }
  //方法,根据指定 URL 字符串获取网络图片资源
  public void getBitmapURLResources(){
     try{
          URL myUrl = new URL(bitmapURL);                  //创建 URL 对象
          URLConnection myConn = myUrl. openConnection();//打开连接
          InputStream in = myConn. getInputStream();       //获得 InputStream 对象
          Bitmap bmp = BitmapFactory. decodeStream(in);    //创建 Bitmap
          ImageView iv = (ImageView)findViewById(R. id. iv); //获得 ImageView 对象
          iv. setImageBitmap(bmp);                         //设置 ImageView 显示的内容
      }
      catch(Exception e){
          e. printStackTrace();
      }
  }
}
```

① 第 21、22 行是定义两个字符串变量，分别指定存放 Tomcat 服务器上 MyUrlFiles 文件夹下的 sgms_msg1.txt 和 sgms_android.png。这里，192.168.1.102 和 8080 是本书案例中 Tomcat 服务器所在计算机的 IP 地址和端口号，读者在学习时可根据自己的计算机的 IP 地址来替换例子中的 IP 地址。在程序中，将这两个文件的地址存放在变量中的好处在于，如果 IP 地址和端口号改变了，只需修改变量定义处，而不必在程序中到处寻找要修改的值。

② 本案例将从 Tomcat 服务器中获得的文本文件和图片文件资源的相关操作分别封装在自定义的方法 getStringURLResources()和 getBitmapURLResources()内，见第 31、32 行。

③ 第 36～55 行，定义从 Tomcat 服务器中获得的文本文件方法 getStringURLResources()。第 39～41 行，首先创建一个 URL 对象 myUrl，该对象指定服务器上的文本文件 sgms_msg1.txt，然后打开 myUrl 的 URL 连接 myConn，从连接中获取输入流 in。使用 URL 获取资源，这是必做的三部曲。接下来由输入流创建一个输入流缓冲区，将数据读入这个缓冲区中，并将读取的数据转换为 UTF-8 格式的字符串，再由 EditText 控件显示出来。

④ 第 57～69 行，定义从 Tomcat 服务器中获得文本文件的方法 getBitmapURLResources()。第 62 行创建一个 Bitmap 对象 bmp，然后由 ImageView 控件将 bmp 显示出来。

(5) 添加权限。打开根目录下的 AndroidManifest.xml，添加网络访问权限，即在 <manifest>标签下添加一条代码：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

【运行结果】 首先将 Eclipse 中的 Tomcat 服务器打开，然后在 Eclipse 的 Android 模拟器上运行 GetURLMsg 项目。初始运行的显示效果如图 10-13 所示。当单击“获取 URL 资源”按钮后，从 Tomcat 服务器中获取文本文件内容和图片文件的内容，并显示出来，效果如图 10-14 所示。




图 10-13 运行的初始界面



图 10-14 单击“获取 URL 资源”按钮后

使用 URL 方式从网络服务器中获取资源的使用方法就这么简单。在获取字符资源时通常需要作格式转换。因为不同的字符集确定不同的字符编码，当然也确定了字符显示的表现。UTF-8 格式是一种被广泛应用的编码，这种编码致力于把全球的语言纳入到一个统一的编码中，目前已经涵盖了几种亚洲语言，如中文、韩文、日文等。

10.2.3 通过 HTTP 获取网络资源

使用 HTTP 请求获取网络资源，包括 POST 和 GET 两种方式。GET 和 POST 是使用 HTTP 的标准协议动词，用于编码和传送变量名/变量值对参数，并且使用相关的请求语义。每个 GET 和 POST 都由一系列 HTTP 请求头组成，这些请求头定义了客户端从服务器请求了什么，而响应则是由一系列 HTTP 应答头和应答数据组成，如果请求成功则返回应答。

但是 GET 和 POST 传送方式不一样，GET 传送的变量名/变量值是作为 URL 的一部分被传送，在 URL 中可以看到，使用这种方式传送的数据量较小，且安全性非常低；而 POST 传送的变量名/变量值则是放在实际的 HTTP 请求消息内部被传送，用户看不到这个过程，使用这种方式传送的数据量较大，且安全性较高。

在 Android 中使用 GET、POST 与网络传送或获取资源需要用到许多位于 org. apache. http 包下的类，其中有两个重要的类是 HttpGet、HttpPost，它们都位于 org. apache. http. client. methods 包下，是用来向网络服务器提交 GET、POST 的请求。使用 HttpGet 或 HttpPost 对象，都必须通过如下三步来访问 HTTP 资源。

(1) 创建 HttpGet 或 HttpPost 对象，将要请求的 URL 通过构造方法传入 HttpGet 或 HttpPost 对象。

(2) 使用 DefaultHttpClient 类的 execute()方法发送 HTTP GET 或 HTTP POST 请求，并返回 HttpResponse 对象。

(3) 通过 HttpResponse 接口的 getEntity()方法返回响应信息，并进行相应的处理。

但是 HTTP GET 请求和 HTTP POST 请求在实际编程中是有区别的，具体代码如下。

1. HTTP GET 请求

```
String url;  //url 中存放网络资源所在的 URL,包含变量名和值
/* 第一步,创建 HttpGet 对象 */
HttpGet get_request = new HttpGet(url);
/* 第二步,使用 execute()方法发送 HTTP GET 请求,并返回 HttpResponse 对象 */
HttpResponse get_response = new DefaultHttpClient().execute(get_request);
if (get_response.getStatusLine().getStatusCode() == 200)   //当状态码为 200 时说明请求成功
{
  /* 第三步,使用 getEntity()方法获得返回结果 */
  String result = EntityUtils.toString(get_response.getEntity());
}
```

2. HTTP POST 请求

```
String url;  //url 中存放网络资源所在的 URL
/* 第一步,创建 HttpPost 对象 */
HttpPost post_request = new HttpPost(url);
/* 设置 HTTP POST 请求参数必须用 NameValuePair 对象 */
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("bookname", etBookName.getText().toString()));
/* 设置 post_request 请求参数 */
post_request.setEntity(new UrlEncodedFormEntity(params, HTTP.UTF_8));
/* 第二步,使用 execute()方法发送 HTTP GET 请求,并返回 HttpResponse 对象 */
HttpResponse post_response = new DefaultHttpClient().execute(post_request);
```

```
if (post_response.getStatusLine().getStatusCode() == 200)
{
  /* 第三步,使用 getEntity()方法获得返回结果 */
  String result = EntityUtils.toString(post_response.getEntity());
}
```

下面通过一个案例来说明使用 GET 和 POST 获取网络资源的用法。

【案例 10.3】 分别通过 GET 和 POST 向网络发送用户名和密码，如果用户名和密码与网络上文件中的信息一致则反馈通过信息，否则反馈不正确信息。

【说明】 通过 GET 和 POST 向网络服务器中发送用户名和密码，使用该网络上的一个文件(例如 http://dev.3gmsc.com/test.php)，来比对传来的用户名和密码是否正确，在该文件中，用户名为“KongFu”，密码也为“KongFu”，并根据比对的结果向客户端返回相应的信息。

本案例没有采用互联网上的获取数据资源的做法，因为无法保证这个文件 test.php 是否永远在 http://dev.3gmsc.com/中，它有可能在某个时候被删除掉。但是在实际应用中，一般是采取这种方式将文件存放于网络服务器上的。作为举例，这里将网络上的文件 test.jsp 存放在本机的 Tomcat 服务器的根目录中(Tomcat 服务器根目录路径为 E:\apache-tomcat-7.0.11\webapps\ROOT)。前面已经提到过，本机的 IP 地址为 192.168.1.102，所以这个文件的 IP 地址为 http://192.168.1.102/test.jsp。注意在 Tomcat 服务器中只能使用 jsp 文件。我们使用这个文件来对上传来的用户名和密码值进行比对，在该文件中，用户名为“KongFu”，密码也为“KongFu”，并完成信息反馈。该文件代码如下所示。

```
1  <%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
2  <%
3  String thisname, thispwd, reqMethod, reqParam;
4  String[] param;
5  thisname = "";
6  thispwd = "";
7  reqMethod = request.getMethod();
8
9  if (reqMethod.equals("GET")) {
10       reqParam = request.getQueryString();
11       if (reqParam != null) {
12             param = reqParam.split("&");
13             thisname = param[0].substring("name".length() + 1);
14             thispwd = param[1].substring("pwd".length() + 1);
15       }
16
17 } else if (reqMethod.equals("POST")) {
18       thisname = request.getParameter("name");
19       thispwd = request.getParameter("pwd");
20 }
21
22 if (thisname.equals("KongFu") &&this pwd.equals("KongFu")) {
23       out.println(reqMethod + " was used. Hello, " + thisname);
24 } else {
25       out.println("Failed to sign in!");
26 }
27
28 %>
```

(1) 第 9～15 行，判断当使用 GET 方式上传时，取上传来的 name 参数的值给变量

thisname，取上传来的 pwd 参数的值给变量 thispwd。

（2）第 17～20 行，当使用 POST 方式上传时，取上传来的 name 参数的值给变量 thisname，取上传来的 pwd 参数的值给变量 thispwd。

（3）第 22～26 行，判断变量 thisname 和 thispwd 的值是否为“KongFu”和“KongFu”，如果一致则返回正确信息，例如使用 GET 方式，则返回信息为“GET was used. Hello，KongFu”；如果不一致则返回信息“Failed to sign in!”。

【开发步骤及解析】

（1）准备文件。将准备好的 test.jsp 文件复制到本机的 Tomcat 文件夹下的 webapps\ROOT 文件夹下，例如本书所用的文件夹为 E:\apache-tomcat-7.0.11\webapps\ROOT。

（2）创建项目。在 Eclipse 中创建一个名为 Get_PostConn 的 Android 项目。其应用程序名为 Get_Post_Conn，包名为 cn.com.sgmsc.httpconn，Activity 组件名为 Get_PostConnActivity。

（3）准备字符串资源。编写 res/layout 目录下的 strings.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>

 <string name = "hello">Hello World, Get_PostConnActivity!</string>
 <string name = "app_name">Get_Post_Conn</string>
 <string name = "local_url">http://192.168.1.102:8080/test.jsp</string>  <!-- 本地机服务器上文件所在地址 -->
 <string name = "remote_url">http://dev.3gmsc.com/test.php</string>  <!-- 远程网络中的文件所在地址 -->

</resources>
```

在这里提供了网络上指定文件的地址信息，供程序使用。

（4）设计布局。编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >

    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal" >

        <TextView
            android:id = "@ + id/userName"
            android:layout_width = "80dip"
            android:layout_height = "wrap_content"
            android:text = "UserName:" />

        <EditText
            android:id = "@ + id/etName"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:ems = "10" >
```

```
24
25              <requestFocus />
26          </EditText>
27
28      </LinearLayout>
29
30      <LinearLayout
31          android:layout_width="fill_parent"
32          android:layout_height="wrap_content">
33
34          <TextView
35              android:id="@+id/tvPasswd"
36              android:layout_width="80dip"
37              android:layout_height="wrap_content"
38              android:text="Password:" />
39
40          <EditText
41              android:id="@+id/etPasswd"
42              android:layout_width="wrap_content"
43              android:layout_height="wrap_content"
44              android:layout_weight="1"
45              android:ems="10"
46              android:inputType="textPassword" />
47
48      </LinearLayout>
49
50      <LinearLayout
51          android:layout_width="fill_parent"
52          android:layout_height="wrap_content"
53          android:gravity="center">
54
55          <Button
56              android:id="@+id/btnGet"
57              android:layout_width="60dip"
58              android:layout_height="wrap_content"
59              android:text="GET" />
60
61          <Button
62              android:id="@+id/btnPost"
63              android:layout_width="60dip"
64              android:layout_height="wrap_content"
65              android:text="POST" />
66
67      </LinearLayout>
68
69  </LinearLayout>
```

① 第 7～28 行定义了一个横向的 LinearLayout 布局，标签<TextView>没有设置 android:layout_weight 属性，那么这个控件占它指定的宽度，标签<EditText>设置 android:layout_weight=1，说明它占余下的宽度。

② 第 23 行，属性 android:ems="10"表示<EditText>的输入宽度为 10 个字符。

③ 第 25 行，<requestFocus />表示这个控件将获得焦点。

④ 第 46 行，属性 android:inputType="textPassword"表示该<EditText>的输入方式

为密码输入方式,即隐藏输入内容。

(5)开发逻辑代码。打开 src/cn.com.sgmsc.httpconn 包下的 Get_PostConnActivity.java 文件,并编辑之。代码如下所示。

```
package cn.com.sgmsc.httpconn;

import java.util.ArrayList;
import java.util.List;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Get_PostConnActivity extends Activity{
 private Button btnGet, btnPost;
 private EditText etName, etPwd;
 private String httpUrl;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        etName = (EditText)findViewById(R.id.etName);
        etPwd = (EditText)findViewById(R.id.etPasswd);
        btnGet = (Button)findViewById(R.id.btnGet);
        btnPost = (Button)findViewById(R.id.btnPost);

        httpUrl = this.getString(R.string.local_url);  //取本地服务器中文件内容的 URL 地址
//       httpUrl = this.getString(R.string.remote_url);//取远程网络的文件内容的 URL 地址

   btnGet.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            HttpGet get_request = new HttpGet(
               String.format("%s?name=%s&pwd=%s",httpUrl,etName.getText(),etPwd.getText())
            );  //根据内容来源地址创建一个 Http 请求
            try {
                /* 发送请求并等待响应 */
                HttpResponse get_response = new DefaultHttpClient().execute(get_request);
                /* 若状态码为 200,说明请求已经成功 */
```

```
52                  if (get_response.getStatusLine().getStatusCode() == 200) {
53                      String get_result = EntityUtils.toString(get_response.getEntity());
                                                            //读返回数据
54                      Toast.makeText(Get_PostConnActivity.this,
55                              get_result,Toast.LENGTH_LONG).show();
56                  }
57              } catch (Exception e) {
58                  Toast.makeText(Get_PostConnActivity.this ,
59                          e.getMessage().toString(), Toast.LENGTH_LONG).show();
60              }
61          }
62      });
63
64      btnPost.setOnClickListener(new OnClickListener() {
65          @Override
66          public void onClick(View v) {
67              HttpPost post_request = new HttpPost(httpUrl);
                                                            //根据内容来源地址创建一个 Http 请求
68              /* 创建一个 ArrayList 对象,向其中添加若干个键-值对 */
69              List<NameValuePair> params = new ArrayList<NameValuePair>();
70              params.add(new BasicNameValuePair("name", etName.getText().toString()));
                                                            //添加键-值对
71              params.add(new BasicNameValuePair("pwd", etPwd.getText().toString()));
72              try {
73                  /* 设置参数的编码 */
74                  post_request.setEntity(new UrlEncodedFormEntity(params,HTTP.UTF_8));
75                  /* 发送请求并获取反馈 */
76                  HttpResponse post_response = new DefaultHttpClient().execute(post_request);
77                  if (post_response.getStatusLine().getStatusCode() == 200) {
78                      String post_result = EntityUtils.toString(post_response.getEntity());
79                      Toast.makeText(Get_PostConnActivity.this,
80                              post_result,Toast.LENGTH_LONG).show();
81                  }
82              } catch (Exception e) {
83                  Toast.makeText(Get_PostConnActivity.this,
84                          e.getMessage().toString(), Toast.LENGTH_LONG).show();
85              }
86          }
87      });
88    }
89 }
```

① 第 2、3 行,引入相关 java.util 的类;第 6～14 行引入相关 org.apache.http 包下的类。

② 第 39、40 行提供了从两个网络上取资源的地址,在本例中使用第 39 行指定的地址,如果想从互联网上获取资源,可把第 39 行改为注释,去掉第 40 行的注释符号,并运行本项目。

③ 第 46 行,使用方法 String.format("%s?name=%s&pwd=%s", httpUrl,etName.getText(), etPwd.getText())得到一个字符串。其被双引号括起来的部分是指定字符串的格式,其后的字符串变量个数和顺序与相应的"%s"对应,即第一个"%s"由 httpUrl 中的值替换,第二个"%s"由 etName.getText()的值替换,第三个"%s"由 etPwd.getText()的值替换。

④ 第 69～71 行是定义一个 ArrayList 对象,向其中添加两个键-值对,为使用 POST 方式传送准备数据。注意这段代码一定要在 onCreate()方法内定义。

(6) 添加权限。打开根目录下的 AndroidManifest.xml,添加网络访问权限,即在

<manifest>标签下添加一条代码：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 Get_PostConn 项目，显示用户名、密码输入界面，这时分别输入“KongFu”和“KongFu”，如图 10-15 所示。如果单击 GET 按钮，可以看到输入信息正确的回馈消息提示，如图 10-16 所示；如果单击 POST 按钮，可以看到如图 10-17 所示的提示信息。



图 10-15　输入了用户名和密码



图 10-16　单击 GET 按钮之后

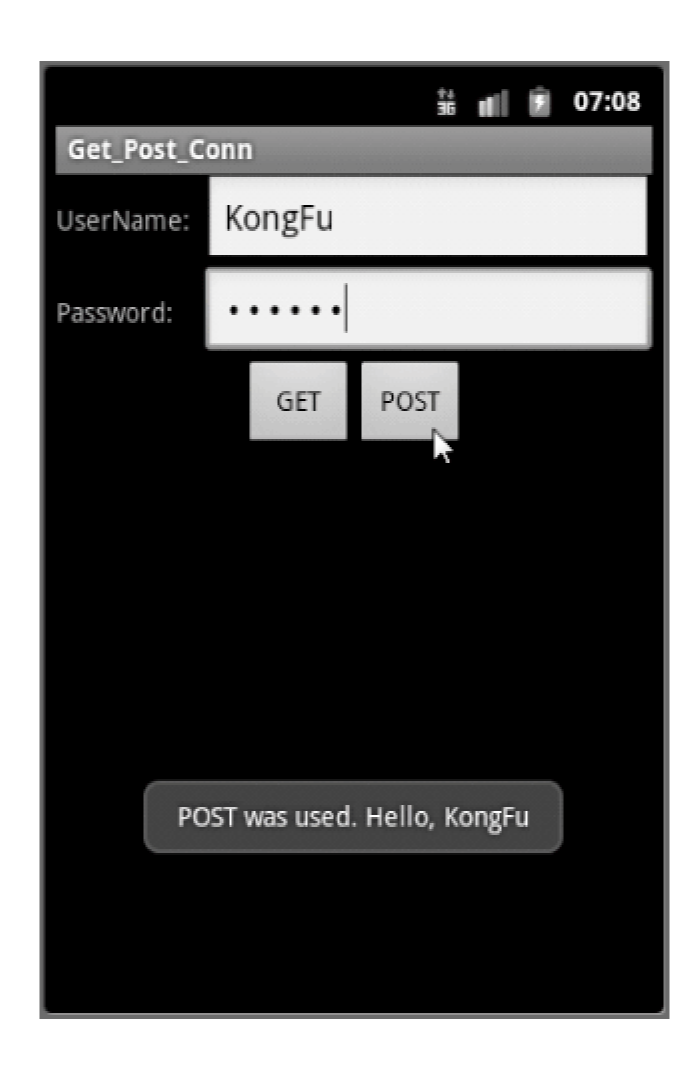


图 10-17　单击 POST 按钮之后

当然，也可以不按正确的用户名和密码输入，看看回馈消息会是什么？

10.3　浏览网页

在 Android 中进行浏览网页的开发非常简单，可使用两种方式，一是使用 Intent 组件，二是使用 WebView 控件。

10.3.1　使用 Intent 组件浏览网页

Intent 是 Android 平台上的一个重要组件，使用它可以调用系统中的 Web 浏览器应用。例如要浏览 Google 中国香港的网页，其关键代码为：

```
Uri uri = Uri.parse("http://www.google.com.hk/");
Intent intent  = new Intent(Intent.ACTION_VIEW,uri);
startActivintenty(intent);
```

下面简单地用一个案例说明其用法。

【案例 10.4】 使用 Intent 组件开发一个浏览网页应用。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 BrowserIntent 的 Android 项目。其应用程序名为 BrowserIntent，包名为 cn.com.sgmsc.BrowserIntent，Activity 组件名为 BrowserIntentActivity。

(2) 准备字符串资源。编写 res/layout 目录下的 strings. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >

    < string name = "app_name"> BrowserIntent </string >
    < string name = "go_button"> Go </string >

</resources >
```

(3) 设计布局。编写 res/layout 目录下的 main. xml 文件，代码如下所示。

```
< LinearLayout
     xmlns:android = "http://schemas.android.com/apk/res/android"
     android:orientation = "horizontal"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent">
     < EditText
         android:id = "@ + id/url_field"
         android:layout_width = "wrap_content"
         android:layout_height = "wrap_content"
         android:layout_weight = "1.0"
         android:lines = "1"
         android:inputType = "textUri"
         android:imeOptions = "actionGo" />
      < Button
         android:id = "@ + id/go_button"
         android:layout_width = "wrap_content"
         android:layout_height = "wrap_content"
         android:text = "@string/go_button" />
</LinearLayout >
```

(4) 开发逻辑代码。打开 src/cn. com. sgmsc. BrowserIntent 包下的 BrowserIntentActivity. java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.BrowserIntent;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnKeyListener;
import android.widget.Button;
import android.widget.EditText;

public class BrowserIntentActivity extends Activity {
    private EditText urlText;
    private Button goButton;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

```

```
23          urlText = (EditText) findViewById(R.id.url_field);
24          goButton = (Button) findViewById(R.id.go_button);
25
26          goButton.setOnClickListener(new OnClickListener() {
27              public void onClick(View view) {
28                  openBrowser();
29              }
30          });
31          urlText.setOnKeyListener(new OnKeyListener() {
32              public boolean onKey(View view, int keyCode, KeyEvent event) {
33                  if (keyCode == KeyEvent.KEYCODE_ENTER) {
34                      openBrowser();
35                      return true;
36                  }
37                  return false;
38              }
39          });
40      }
41
42      /* 打开 EditView 中指定 URL 的网页   */
43      private void openBrowser() {
44          Uri uri = Uri.parse(urlText.getText().toString());
45          Intent intent = new Intent(Intent.ACTION_VIEW, uri);
46          startActivity(intent);
47      }
48  }
```

① 第 26～30 行，为按钮添加一个 OnClickListener()监听，第 31～39 行，为 EditView 的控件 urlText 添加一个 OnKeyListener()监听，在两个监听的回调方法中都调用了 openBrowser()方法，在该方法中编写打开网页的代码。

② 第 33 行，if (keyCode == KeyEvent.KEYCODE_ENTER)是判断按下的键是否为回车键，只有在按下回车键时才能调用 openBrowser()方法。

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 BrowserIntent 项目，在文本编辑框中可以输入完整的网址，如图 10-18 所示，当单击 Go 按钮后，或者在输入框中按回车键后，打开该网页，如图 10-19 所示。

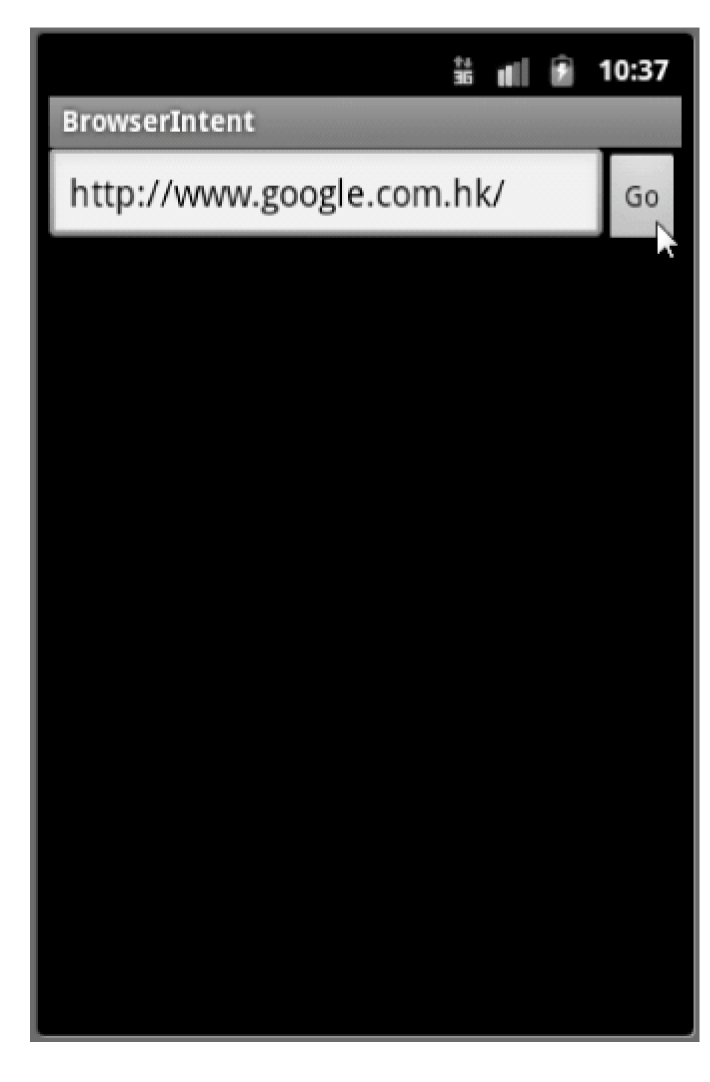


图 10-18 在编辑框内输入网址



图 10-19 打开输入网址的网页

10.3.2 使用 WebView 控件浏览网页

WebView 类位于 android. webkit 包下，它可以对网页浏览和控制进行设置，常用的方法如表 10-1 所示。

表 10-1 WebView 类常用的方法及说明

方　法	描　述
setJavaScriptEnabled(Boolean flag)	设置网页中是否支持 JavaScript。如果 flag 为 true，则允许 WebView 可以执行 JavaScript
setHorizontalScrollBarEnabled(Boolean enable)	设置水平滚动条的显示/隐藏。如果 enable 为 true，则显示水平滚动条
loadUrl (Url url)	打开 url 指定的网页
goBack()	向后显示浏览过的页面
goForward()	向前显示浏览过的页面

下面使用一个简单案例来说明 WebView 的用法。

【案例 10.5】 使用 WebView 控件开发一个浏览网页应用。

【说明】 使用 WebView 开发浏览网页应用时，需要在 AndroidManifest. xml 中添加网络权限。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 BrowserView 的 Android 项目。其应用程序名为 BrowserView，包名为 cn. com. sgmsc. BrowserView，Activity 组件名为 BrowserViewActivity。

(2) 准备字符串资源。编写 res/layout 目录下的 strings. xml 文件，该文件与案例 10. 4 中的 strings. xml 文件内容相似，在此省略其代码显示。

(3) 设计布局。编写 res/layout 目录下的 main. xml 文件，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <EditText
            android:id="@+id/url_field"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:lines="1"
            android:inputType="textUri"
            android:imeOptions="actionGo" />
        <Button
            android:id="@+id/go_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/go_button" />
```

```
24         </LinearLayout>
25         <WebView
26             android:id="@+id/web_view"
27             android:layout_width="fill_parent"
28             android:layout_height="wrap_content"
29             android:layout_weight="1.0" />
30     </LinearLayout>
```

在布局代码中要添加一个<WebView>标签，见第25～29行。

(4) 开发逻辑代码。打开 src/cn.com.sgmsc.BrowserView 包下的 BrowserViewActivity.java 文件，并编辑之。代码如下所示。

```
package cn.com.sgmsc.BrowserView;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnKeyListener;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.EditText;

public class BrowserViewActivity extends Activity {
    private EditText urlText;
    private Button goButton;
    private WebView webView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        urlText = (EditText) findViewById(R.id.url_field);
        goButton = (Button) findViewById(R.id.go_button);
        webView = (WebView) findViewById(R.id.web_view);

        goButton.setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                openBrowser();
            }
        });
        urlText.setOnKeyListener(new OnKeyListener() {
            public boolean onKey(View view, int keyCode, KeyEvent event) {
                if (keyCode == KeyEvent.KEYCODE_ENTER) {
                    openBrowser();
                    return true;
                }
                return false;
            }
        });
    }

    /* 打开 EditView 中输入网址的网页 */
    private void openBrowser() {
        webView.getSettings().setJavaScriptEnabled(true);
```

```
46          webView.loadUrl(urlText.getText().toString());
47      }
48  }
```

① 该逻辑代码的编程思想与案例 10.4 的相似，不同之处在于打开网页的自定义方法 openBrowser()。

② 第 45 行，设置 WebView 对象可以访问页面中有 JavaScript 的网页。现在的网页一般都会使用到 JavaScript 脚本语句编写。

(5) 添加权限。打开根目录下的 AndroidManifest.xml，添加网络访问权限，即在<manifest>标签中添加一条代码：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

【运行结果】 在 Eclipse 中启动 Android 模拟器，然后运行 BrowserView 项目，在文本编辑框中可以输入完整的网址，如图 10-20 所示，当单击 Go 按钮后，或者在输入框中按回车键后，打开该网页，如图 10-21 所示。

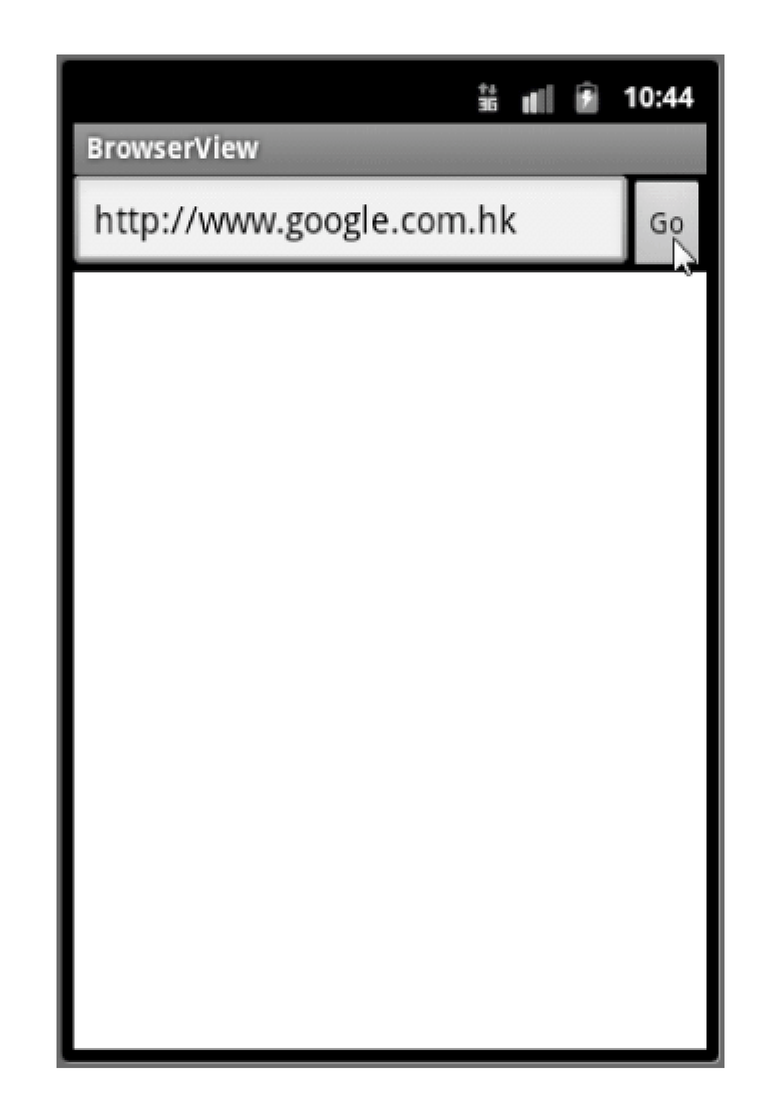


图 10-20　在编辑框内输入网址



图 10-21　打开输入网址的网页

读者可以比较一下这两个案例的运行界面，看看有什么区别？注意，如果在应用程序中使用 WebView 对象多了，会影响到程序的执行效率。

10.4　定位与 Google 地图

Google 公司早在 2005 年就推出了 Google Maps 等基于位置的服务，Android 出自 Google 之手，那么将诸如 GPS 定位、Google Map、天气预报等多种服务应用到 Android 中是自然不在话下的。

10.4.1　Google 位置服务

在 Android 的位置服务中，有两个重要的类是 LocationManager 和 LocationProvider。

LocationManager 类位于 android.location 包下，它提供了用于访问设备位置信息的服务，这些服务可以使应用程序周期性地获得设备的位置数据，还可以在设备的地理位置满足特定条件时触发 Intent 广播。LocationProvider 类定义了位置服务的提供方法，例如，是由 GPS 设备提供还是通过网络提供等，其中静态字符串常量 GPS_PROVIDER 表示 LocationProvider 是 GPS，静态字符串常量 NETWORK_PROVIDER 表示 LocationProvider 是网络。

1. LocationManager 及相关类的方法

LocationManager 类的对象通过 Context.getSystemService(Context.LOCATION_SERVICE)方法获得实例。一旦获得 LocationManager 的实例，就可以实现获取设备的位置、周期性地更新位置等功能。该类常用的方法如表 10-2 所示。

表 10-2　LocationManager 常用的方法及说明

方　法	描　述
addGpsStatusListener(GpsStatus.Listener listener)	添加一个 GPS 状态监听器
getAllProviders()	获得所有的 LocationProvider 列表
getBestProvider(Criteria criteria, boolean enabledOnly)	传入 Criteria 对象，返回与 Criteria 对象设置条件最匹配的 LocationProvider，作为 getLastKnownLocation 方法的传入参数
getLastKnownLocation (String provider)	根据 Provider 获得位置信息，其默认的 Provider 是 GPS。该方法返回一个封装了经纬度等信息的 Location 对象
getProvider(String name)	获得指定名称的 LocationProvider
requestLocationUpdates(String provider, long minTime, float minDistance, PendingIntent intent)	通过给定的 Provider 名称，周期性地通知当前的 Activity。其中 minTime 和 minDistance 代表地理位置更新的最小时间间隔及位移变化的最短距离
requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener listener)	添加一个 LocationListener 监听器。其中 provider 为注册的 provider 名
removeUpdate(LocationListener listener)	移除指定的 LocationListener 监听器

方法中涉及一个 Criteria 类，该类表示了应用程序选择位置服务 Provider 的一个标准。Provider 可能是根据精准度，电量使用，能否获得海拔，速度、方向和产生资费来选择的。Criteria 类提供了一系列的 set()方法，可以设置多种因素的标准，LocationManager 可以根据这个设定好的标准，自动选择最适合需求的 Provider。Criteria 类常用的方法如表 10-3 所示。

表 10-3　Criteria 常用的方法及说明

方　法	描　述
setAccuracy(int accuracy)	设置经纬度的精准度。可选参数有 ACCURACY_FINE(准确)，ACCURACY_COARSE(粗略)
setAltitudeRequired(boolean altitudeRequired)	设置是否需要获取海拔数据
setBearingAccuracy(int accuracy)	设置方向的精确。可选参数有 ACCURACY_LOW(低)，ACCURACY_MEDIUM(中)，ACCURACY_HIGH(高)，NO_REQUIREMENT(没有要求)
setBearingRequired(boolean bearingRequired)	设置是否需要获得方向信息
setCostAllowed(boolean costAllowed)	设置是否允许定位过程中产生资费，例如流量等

续表

方　法	描　述
setPowerRequirement(int level)	设置耗电量的级别。可选参数有 POWER_LOW(低)，POWERMEDIUM(中)，POWER_HIGH(高)，NO_REQUIREMENT(没有要求)
setSpeedAccuracy(int accuracy)	设置速度的精确度
setSpeedRequired(boolean speedRequired)	设置是否提供速度的要求

2. 添加位置变化监听器

调用 LocationManager 类的 getLastKnownLocation()方法只是主动地查询地理位置信息，如果需要在地理位置信息发生变化后自动通知系统，可以为 LocationManager 添加一个 LocationListener 监听器。

通过调用 LocationManager 的 requestLocationUpdates()方法可以添加一个 LocationListener 监听器，调用 LocationManager 的 removeUpdates()方法可以移除指定的 LocationListener。

在 LocationListener 监听器中需要实现下列几个方法，分别描述如下。

(1) onLocationChanged(Location location)，当设备位置信息发生变化时调用该方法。

(2) onProviderDisabled(String provider)，当设备的 Location Provider 被禁用时调用该方法。如果注册监听器时设备已经禁用了 Location Provider，则会立即调用该方法。

(3) onProviderEnabled(String provider)，当设备的 Location Provider 被启用时调用该方法。

(4) onStatusChanged(String provider, int status, Bundle extras)，当设备的 Location Provider 状态发生变化时触发该方法，可取的状态为 TEMPORARILY_UNAVAILABLE、OUT_OF_SERVICE 和 AVAILABLE。

3. 地理位置信息定位应用开发中的几点注意

在开发地理位置定位应用时需要注意以下几个问题。

1) 添加相关的权限

使用 GPS 卫星进行定位，需要在 AndroidManifest.xml 文件中的<manifest>标签下添加获取位置地理信息的权限。根据不同的 LocationProvider，添加的权限也有些细微不同。

(1) GPS 定位。

使用 GPS 卫星进行定位，需要添加权限 android.permission.ACCESS_FINE_LOCATION。

(2) NETWORK 定位。

使用信号接收塔和 WIFI 介入点进行定位，需要添加权限 android.permission.ACCESS_FINE_LOCATION(精确定位)或 android.permission.ACCESS_COARSE_LOCATION(粗糙定位)。

2) Android SDK 的版本问题

对 GPS 管理的功能模块，在 Android SDK 版本不同的情况下，对 GPS 控制的代码是不一样的。在 SDK 2.2 及以下的版本中，可以利用 Android 平台自带的 Widget 插件对各种开关进行管理，从而实现 GPS 的开关。使用前面介绍的方法来编写代码，可以正常运行。

但在 2.3 版本里面，需要利用 SDK 中的类 Settings.Secure 的一个静态方法，那么在

AndroidManifest.xml 中需要添加“android.permission.WRITE_SECURE_SETTINGS”权限，而 Google 把这个权限完全锁住了。如果想在 SDK2.3 版本管理 GPS，那么只能使用 Intent 打开系统默认的管理 GPS 的 Activity 了。

3）在模拟器上模拟 GPS 位置数据

Google 位置服务应用只有在接通网络的移动设备上才真正有意义，如 Android 手机，平板计算机等。而在开发中使用的模拟器是没有 GPS 设备的，在运行应用程序时需要模拟 GPS 位置数据。

设置 Eclipse 的模拟器 GPS 位置的方法有两种，一种是使用 DDMS 工具提供的 Emulator Control，另一种是在命令行状态使用命令。

（1）使用 DDMS 工具的操作步骤。

首先启动 Eclipse 的模拟器；然后进入 Eclipse 的 DDMS 工具，选择菜单 Window→Show View→Other...，打开 Show View 对话框；选择 Android 下的 Emulator Control，打开 Emulator Control，在 Location Controls 的 Manual 选项卡中输入模拟的地理位置经纬度，这里，默认选择 Decimal（十进制），也可以选择 Sexagesimal（六十进制），Longitude 是经度，Latitude 是纬度，可以在输入框中输入一对经、纬度数值，如图 10-22 所示；然后单击 Send 按钮即完成模拟器的 GPS 位置定位。

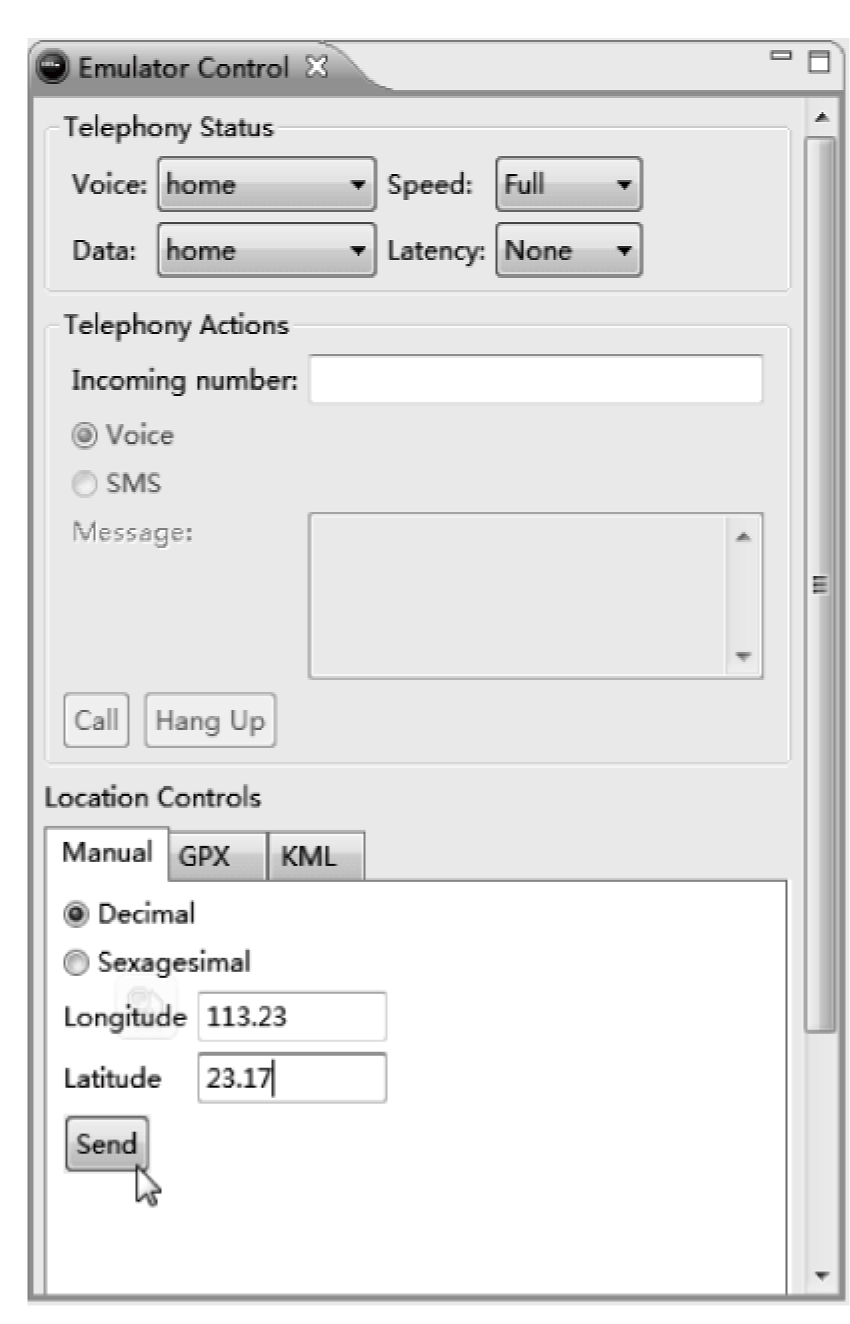


图 10-22　Emulator Control 对话框

（2）使用命令的操作方法。

首先进入命令行状态（在 Windows 中运行 cmd 可执行文件），然后输入下列命令：

```
telnet localhost 5554
```

该命令是打开模拟器，其中 5554 是模拟器在本机的端口，不同的模拟器端口号不同，如果在 Eclipse 中打开两个模拟器，通常第一个模拟器是 5554，第二个模拟器是 5556。这个端口号显示在模拟器左上方，如果模拟器端口号不同，在输入命令时请使用自己的模拟器端口号。执行了该命令后进入 Android 控制台状态，如图 10-23 所示。

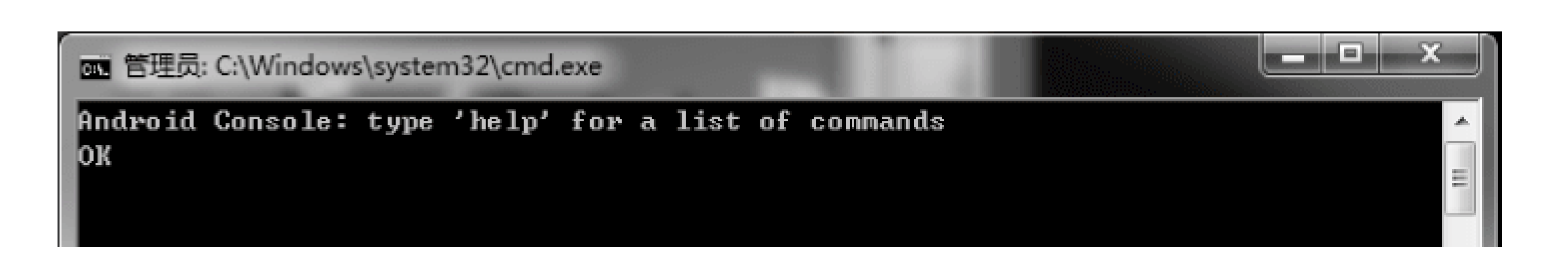


图 10-23　输入了 telnet 命令之后进入 Android Console

在 Android 控制台状态输入下列命令。

```
geo fix 113.23 23.17
```

命令的格式为：geo fix 经度 纬度，用于设置 Android 模拟器位置的经度和纬度。

注意，如果是使用 Windows 7 的用户，控制台可能会提示 telnet 之类的信息，因为 Windows 7 系统下默认是不出现 telnet 的，需要手动打开。具体操作步骤如下。

① 在控制面板中，选择“程序”，单击“打开或关闭 Windows 功能”，进入“Windows 功能”对话框，将“Telnet 服务器”和“Telnet 客户端”勾选上，如图 10-24 所示。



图 10-24 “Windows 功能”对话框

② 仍在控制面板里，打开“系统和安全”，然后再打开“管理工具”，双击“服务”，进入“服务”窗口，将 Telnet 项的启动类型由“禁用”改为“手动”，完成后如图 10-25 所示。

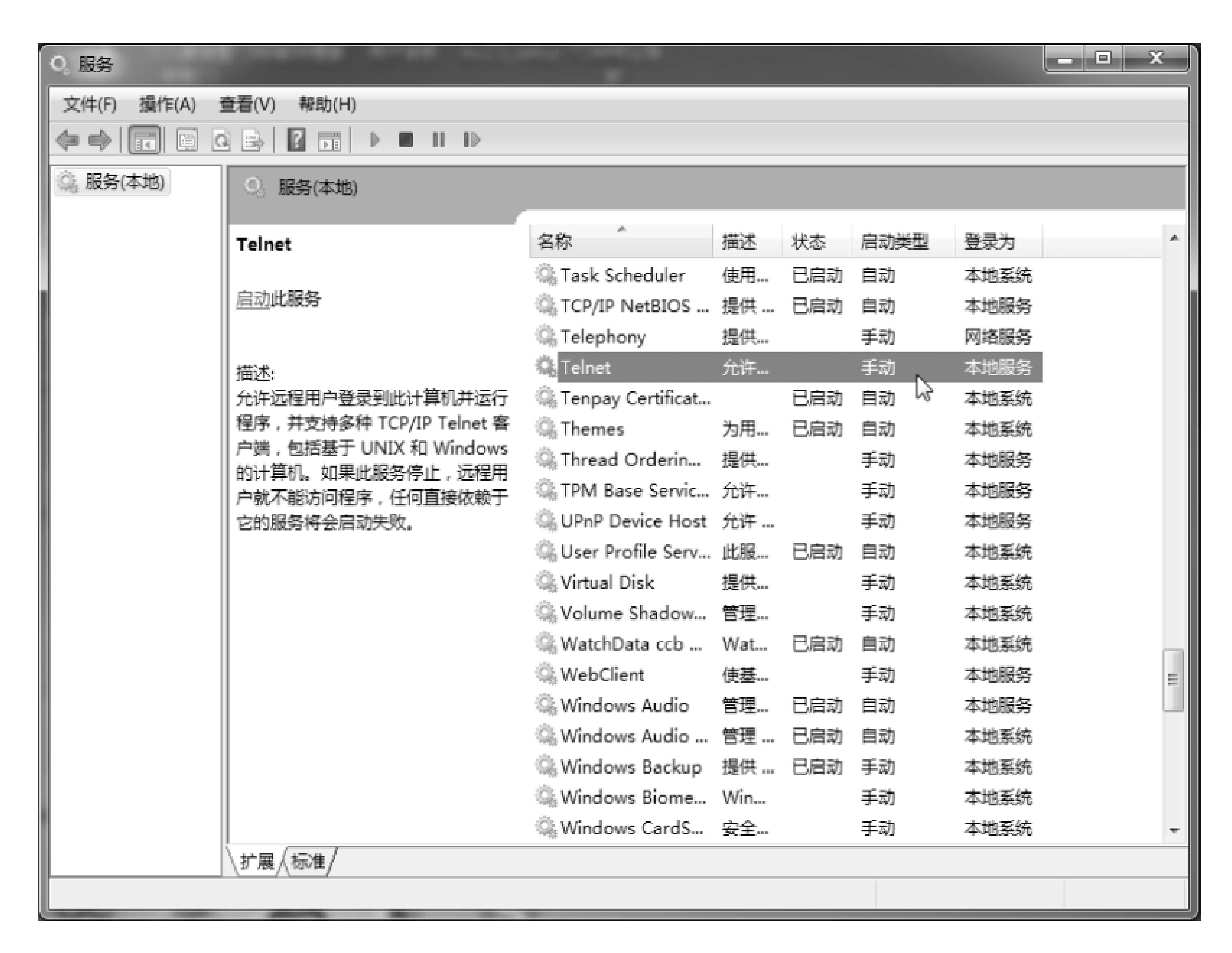


图 10-25 “服务”窗口

③ 完成上述操作之后，就可以在命令行状态下输入 telnet localhost 命令和 geo fix 命令了。

4. 中国几个城市的经纬度

表 10-4 中列出了中国几个大中城市的经、纬度，这些经纬度数据存在细小偏差，仅供参考。

表 10-4 中国几个城市的经纬度参考数据

城市	经度	纬度	城市	经度	纬度
北京	116:28E	39:54N	青岛	120:19E	36:04N
天津	117:10E	39:10N	郑州	113:42E	34:44N
石家庄	114:26E	38:03N	开封	114:23E	34:52N
上海	121:26E	31:12N	泸州	105:27E	28:54N
南京	118:46E	32:03N	万县	108:22E	30:48N
杭州	120:10E	30:15N	常德	111:39E	29:00N
宁波	121:34E	29:53N	广州	113:18E	23:10N

5. GPS 定位简单应用

【案例 10.6】 列出你手机所在的位置信息。

【说明】 前面已经提到了关于 SDK 的版本问题。所以在本例中使用 API Level 级别为 8 的模拟器来运行程序,因为 API Level 8 对应的 SDK 版本是 2.2。否则可能得不到预期运行结果。

在开发 GPS 定位应用时要注意所使用的 Android SDK 版本,由于版本的不同,在编程中对 GPS 控制的代码也不一样。希望读者要注意到这些细节。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 MyLocation 的 Android 项目。其应用程序名为 LocationDemo,选择 Build Target 时勾选 Android 2.2,包名为 cn.com.sgmsc.Location,Activity 组件名为 LocationDemoActivity。

(2) 设计布局。编写 res/layout 目录下的 main.xml 文件,代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:orientation = "vertical"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      >
7  <EditText
8      android:id = "@ + id/et"
9      android:layout_width = "fill_parent"
10     android:layout_height = "wrap_content"
11     android:cursorVisible = "false"
12     android:editable = "false"
13     />
14 </LinearLayout>
```

第 11 行定义该 EditText 控件内的光标不可见。第 12 行定义该 EditText 控件不可编辑。定义了这两个属性,那么这个 EditText 只能用于显示信息。

(3) 开发逻辑代码。打开 src/cn.com.sgmsc.Location 包下的 Service_BdcastActivity.java 文件,并编辑之。其代码如下所示。

```
package cn.com.sgmsc.Location;

import android.app.Activity;
```

```
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.provider.Settings;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.widget.EditText;
import android.widget.Toast;

public class LocationDemoActivity extends Activity {
 LocationManager lm;                                    //声明 LocationManager 对象的引用
 EditText et;                                           //声明 EditText 对象的引用
 LocationListener llistener = new LocationListener(){   //声明一个 LocationListener 对象
     @Override
     public void onLocationChanged(Location location) {//重写 onLocationChanged()方法
         updateView(location);
     }
     @Override
     public void onProviderDisabled(String provider) { //重写 onProviderDisabled()方法
         updateView(null);
     }
     @Override
     public void onProviderEnabled(String provider) {  //重写 onProviderEnabled()方法
         Location l = lm.getLastKnownLocation(provider);           //获取位置信息
         updateView(l);                                 //更新 EditText 控件的内容
     }
     @Override
     public void onStatusChanged(String provider, int status, Bundle extras) {
                                                        //重写 onStatusChanged()方法
     }
 };
    @Override
    public void onCreate(Bundle savedInstanceState) {  //重写 onCreate()方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                  //设置当前屏幕

        et = (EditText)findViewById(R.id.et);           //获得 EditText 对象
        openGPSSettings();        //调用打开 LocationManager()方法,并判断 GPS 模块是否开启
        getLocation();                                  //调用获取地理位置信息方法
    }

    //方法:判断 GPS 模块是否存在或者是开启
    private void openGPSSettings() {
        LocationManager alm = (LocationManager) this
                .getSystemService(Context.LOCATION_SERVICE);
        if (alm
                .isProviderEnabled(android.location.LocationManager.GPS_PROVIDER)) {
            Toast.makeText(this, "GPS 模块正常", Toast.LENGTH_SHORT)
                    .show();
            return;
        }
        Toast.makeText(this, "请开启 GPS!", Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(Settings.ACTION_SECURITY_SETTINGS);
```

```
58          startActivityForResult(intent,0);           //此为设置完成后返回到获取界面
59      }
60                                                      //方法：获取到地理位置信息
61      private void getLocation()
62      {
63          /* 获取位置管理服务 */
64          LocationManager lm = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
65          String provider = lm.getBestProvider(getCriteria(), true);        //获取 GPS 信息
66          Location location = lm.getLastKnownLocation(provider);          //通过 GPS 获取位置
67          updateView(location);                           //更新 EditText 控件的内容
68          /* 设置监听器,自动更新的最小时间为间隔 N 秒(1 秒为 1 * 1000,这样写主要为了方便)
69           * 和最小位移变化超过 10 米 */
70          lm.requestLocationUpdates(provider, 5 * 1000, 10, llistener);
71      }
72      //方法：返回查询条件
73      public Criteria getCriteria(){
74          Criteria c = new Criteria();
75          c.setAccuracy(Criteria.ACCURACY_FINE);          //设置查询精度为高精度
76          c.setSpeedRequired(false);                      //设置是否要求速度
77          c.setCostAllowed(false);                        //设置是否允许产生费用
78          c.setBearingRequired(false);                    //设置是否需要得到方向
79          c.setAltitudeRequired(false);                   //设置是否需要得到海拔高度
80          c.setPowerRequirement(Criteria.POWER_LOW);      //设置允许的电池消耗级别为低功耗
81          return c;                                       //返回查询条件
82      }
83      //方法：更新 EditText 中显示的内容
84      public void updateView(Location newLocation){
85          if(newLocation != null){                        //判断是否为空
86              et.setText("您现在的位置是\n 经度：");
87              et.append(String.valueOf(newLocation.getLongitude()));     //获得精度
88              et.append("\n 纬度：");
89              et.append(String.valueOf(newLocation.getLatitude()));     //获得纬度
90          }
91          else{   //如果传入的 Location 对象为空则清空 EditText
92              et.getEditableText().clear();               //清空 EditText 对象
93          }
94      }
95
96  }
```

① 第 18～35 行，声明并创建一个 LocationListener 对象 llistener，在其中重写了 onLocationChanged()、onProviderDisabled()、onProviderEnabled()和 onStatusChanged() 4 个方法。

② 第 42 行调用 openGPSSettings()方法，该方法定义在第 47～59 行。其中创建了 LocationManager 对象 alm，指定 LocationProvider 类型为 GPS_PROVIDER，即使用 GPS 定位；并判断 GPS 模块是否开启。第 57 行，使用了 Settings. ACTION_SECURITY_SETTINGS 创建一个 Intent 对象，这里的 Settings 是一个系统属性设置类，它位于 android.provider 包下，有许多的系统属性都是在 Settings 应用当中进行设置的，例如 GPS、WIFI、蓝牙状态、当前本机语言及屏幕亮度等一些相关的系统属性值。

③ 第 43 行调用 getLocation()方法，该方法定义在第 61～71 行。其中第 65 行是通过 lm. getBestProvider(getCriteria(), true)获得地理位置信息的提供者，Criteria 对象由

getCriteria()方法得到。第 66 行通过 GPS 获取地理位置信息。第 67 行将所得信息显示在 EditText 控件中。第 70 行设置监听器，定义每间隔 5s 或每移动 10m 执行一次提取经纬度数据。

（4）确定 SDK 版本，添加权限。打开根目录下的 AndroidManifest.xml，将 minSdkVersion 的值设置为“8”，添加网络访问权限。在默认生成的 AndroidManifest.xml 文件中，其<manifest>标签内应该有下列代码：

```
<uses-sdk android:minSdkVersion="8" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

【运行结果】 在 Eclipse 中启动 API Level 8 的 Android 模拟器，然后运行 MyLocation 项目。如果没有设置模拟器的 GPS 数据，运行界面没有任何位置信息。如果按本节介绍的方法设置了模拟器的 GPS 的经纬度数据，则界面显示如图 10-26 所示。

图 10-26 显示模拟器的模拟 GPS 经纬度数据

10.4.2 Google Map 应用

Google Map 是 Google 公司提供的电子地图服务，包括局部详细的卫星照片。它能提供以下三种视图。

（1）矢量地图，即为传统地图。这类地图可提供行政辖区和交通以及商业信息。

（2）卫星照片，即为俯视地图。这类地图与 Google Earth 工具软件显示的卫星照片一样，可以看到一些地区的实景，如世界各地的城市街区、重要基地等。

（3）地形视图。可以显示地形和等高线的地图。

为了使得应用程序的地图功能更加强大，Google 提供一个 Maps 外部函数库，包含在 com.google.android.maps 包内。该包中的类提供有内置的下载、渲染和地图拼接的缓冲，以及各种显示选项和控制。其中其关键类是 MapView，它是 ViewGroup 类的子类，为用户提供所有必要的 UI 元素以控制地图，例如当 MapView 获得焦点时，它可以捕捉按键、触摸手势，自动地平移和缩放地图等。

要在 MapView 中显示 Google 地图数据，必须注册 Google Maps 服务，并获得一个 Maps API Key。

1. 申请 Map API Key

要在 MapView 组件中显示 Google Maps 数据，就必须注册一个 Google Maps API Key。每个 Maps API Key 是唯一与一个特定的基于 MD5 指纹的证书联系在一起的，并且申请 Android Map API Key 还需要一个 Google 的账号。如果没有 Google 的账号可以先上 google 网站去申请一个账户。

在开发 Google Map 服务的应用时，程序员必须先申请一组验证过的 Map API Key。为了能顺利地申请 Android Map API Key，又必须要准备一个 Google 的账号和系统的数字证书（即 MD5 指纹）。申请 Map API Key 的过程如下。

1）申请 Google 账号

在 IE 中输入 Google 网址：http://www.google.com.hk/，然后单击“登录”，按照提示注

册一个 Google 的账号。这个 Google 账号是通用的，可以用来注册 Map API Key。请记住申请 Google 账号时使用的邮箱和 Google 账号密码。

2）获取 Map API Key

在较早时期，Google Maps API 是免费提供的。如果应用程序要使用到 Google Map，就必须获取一个 Map API Key。要注册一个 Maps API Key，需要提供一个用来对应用程序进行签名的证书的 MD5 指纹。从 2012 年之后，Google Maps API 不再免费提供，当用户调用 Google Maps API 每天超过一定限制次数（目前限定于 25 000 次）之后，会按照超出的次数来收取费用，费用是每一千次调用 0.5 美元左右。在 Google Maps API 实施部分收费之后，Google 提供了 APIs Console 进行所有 API 的管理，但仍支持使用 Map API Key。

（1）MD5 指纹。

MD5 的全称是 message-digest algorithm 5，即信息摘要算法，被广泛用于加密和解密技术上，它又称为文件的“数字指纹”。如同人的指纹一样，任何人都有自己独一无二的指纹，与之类似，MD5 可以为任何文件（不管其大小、格式、数量）产生一个同样独一无二的“数字指纹”，如果任何人对文件做了任何改动，其 MD5 值也就是对应的“数字指纹”都会发生变化。要注册一个 Maps API Key，需要提供一个用来对应用程序进行签名的证书的 MD5 指纹。生成 MD5 指纹的操作步骤如下。

首先查看 Eclipse 中的 Default debug keystore 的路径。选择菜单 Window→Preferences，进入 Preferences 对话框，展开左侧窗口的 Android，单击 Build 项，记住右侧窗口中的 Default debug keystore 的路径，如图 10-27 所示。

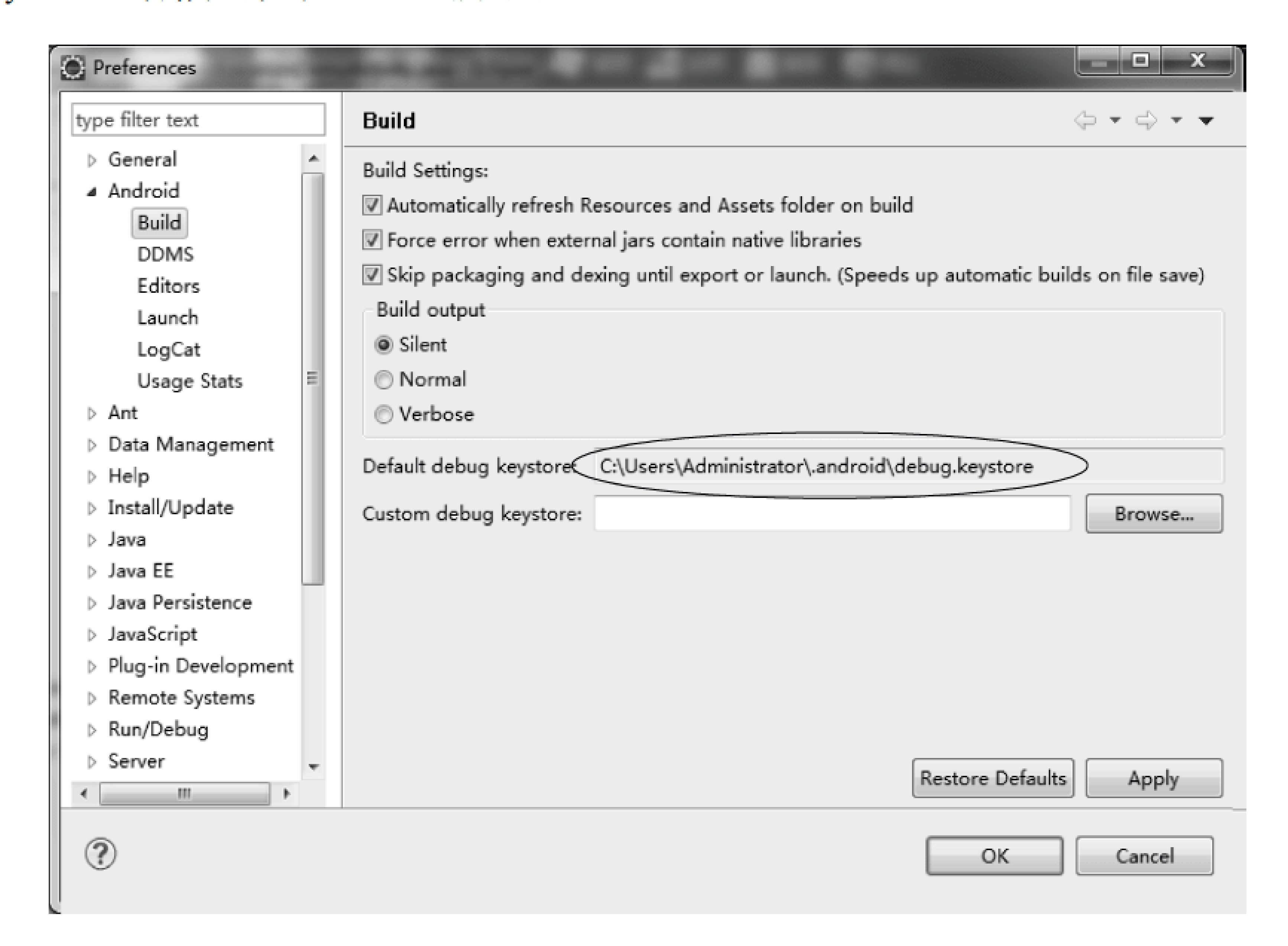


图 10-27　Preferences 对话框

然后在命令行状态（运行 cmd 进入命令行状态）下，使用 JDK 自带的 keytool 工具，输入命令：

```
Keytool -list -alias androiddebugkey -keystore "C:\Users\Administrator\.android\debug.keystore" -storepass android -keypass android
```

如图 10-28 所示，输入完此命令之后回车，即可得到一个 MD5 指纹，如图 10-29 所示。



图 10-28 输入 keytool... 命令



图 10-29 得到 MD5 认证指纹

这里得到的 MD5 认证指纹为 1C:7D:3F:D9:AF:56:6B:58:2B:EA:DA:13:F5:9A:A6:D7:79:50:BF:47。请注意，不同的应用程序其 MD5 指纹不同，可以使用上述操作方法得到自己的 MD5 认证指纹。这个 MD5 认证指纹很重要，一个应用程序发布时必须要有 MD5 指纹。所以在开发时要将自己的 MD5 指纹妥善保管好。

(2) 申请 Map API Key 方式。

从 2012 年 2 月开始，Google 变更了其 Maps API 的申请方式，不再提供以前的申请 API Key 方式。下面介绍新的 Map API Key 申请方式。

在浏览器的地址栏中输入如下网址：https://code.google.com/apis/console，使用 Google 账号登入 APIs Console，如图 10-30 所示。

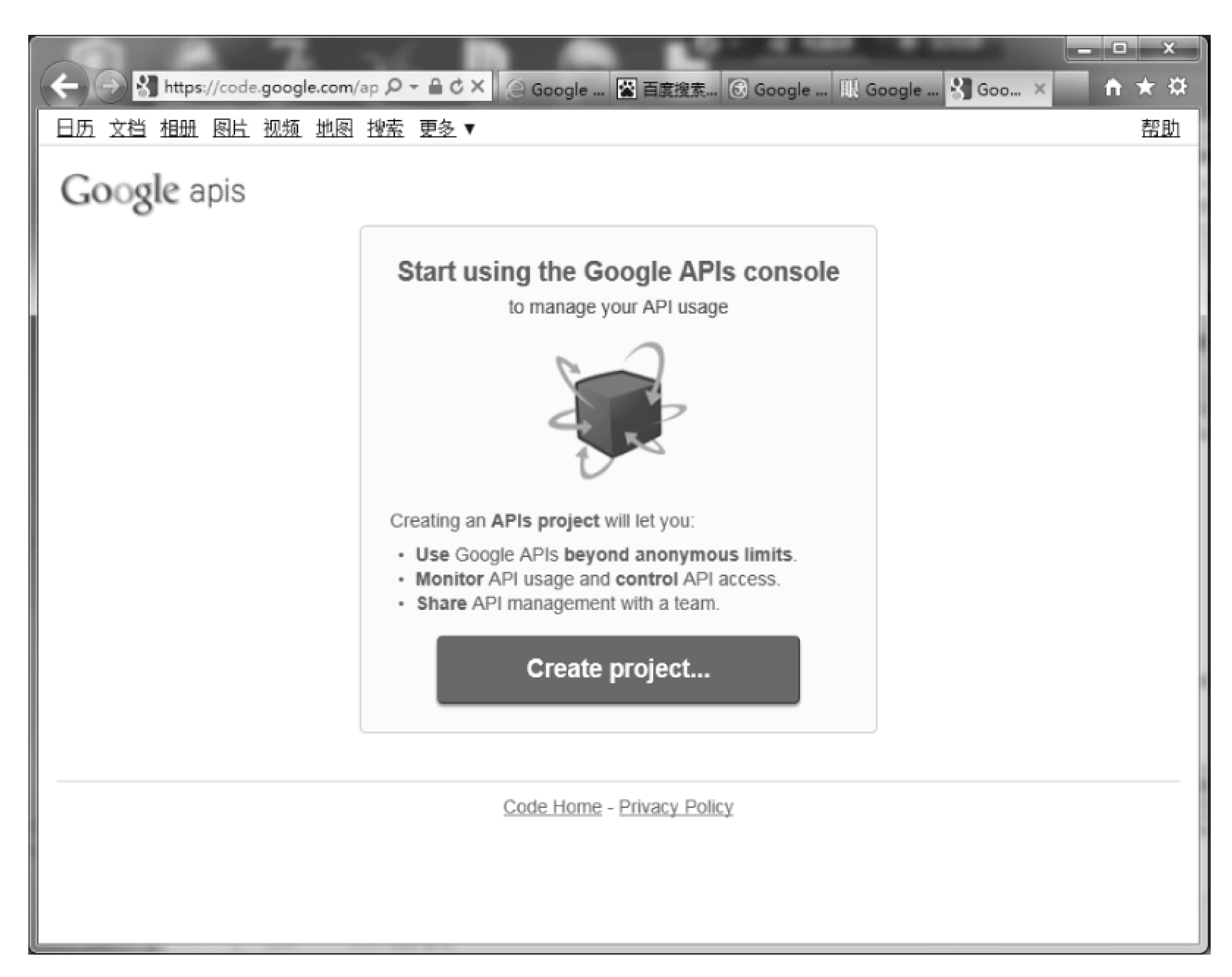


图 10-30 Google APIs console 的进入界面

单击 Create project...按钮，进入下一页面，在左侧的 API Project 下单击 Servers，在右侧窗口显示所有的 Servers 列表。向下找到 Google Maps API v2 项，将其 Status（状态）设置为 On，如图 10-31 所示。

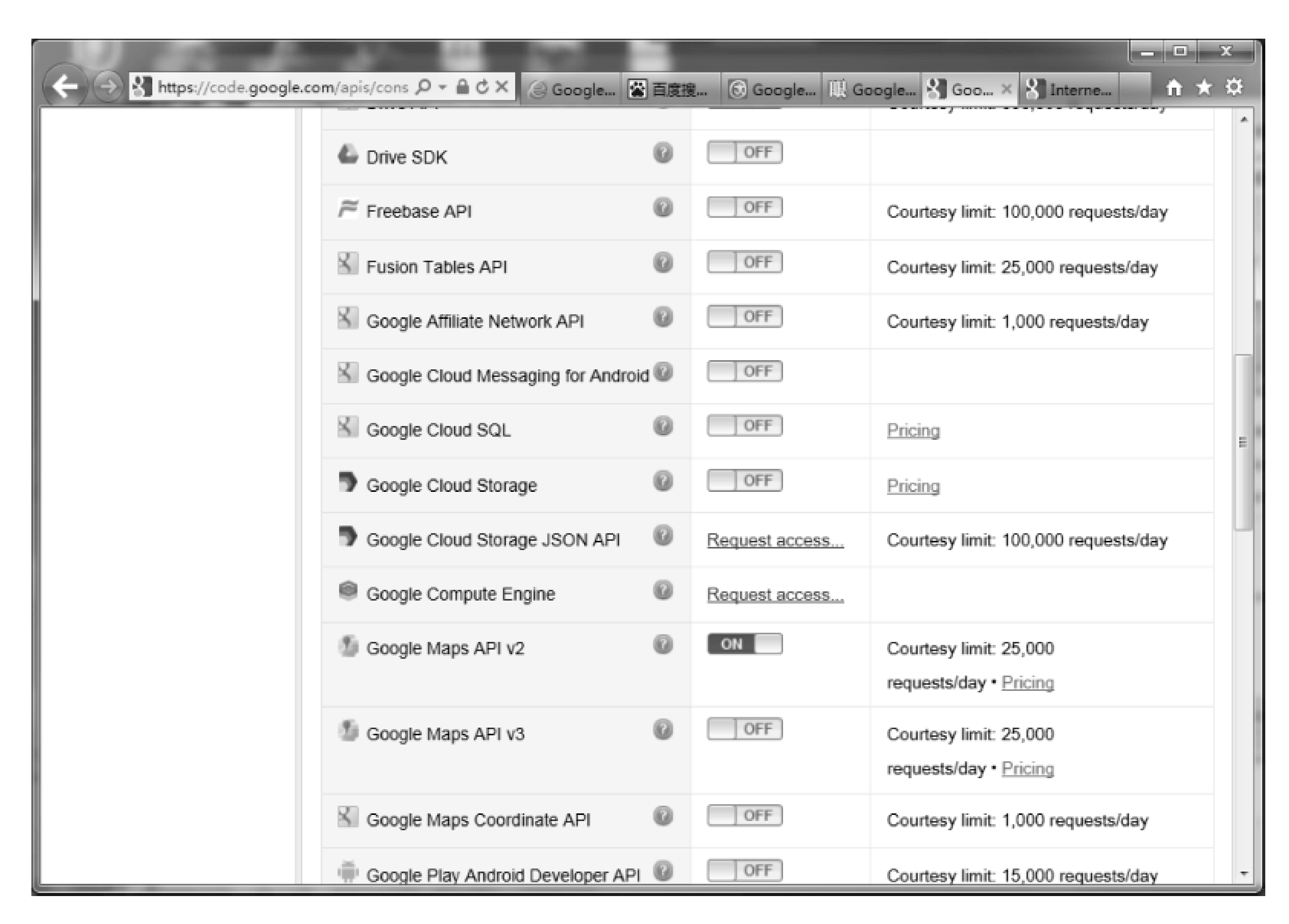


图 10-31　对 API Project 的 Servers 列表项的状态设置

在左侧的 API Project 下单击 API Access，在右侧窗口单击 Create new Browser key...，然后填入要生成 Key 的网址，单击 Create 按钮，即可得到一个 API Key，如图 10-32 所示。如果已经有一个旧的 Key，可以单击右侧的 Generate new key...生成新的 API Key。

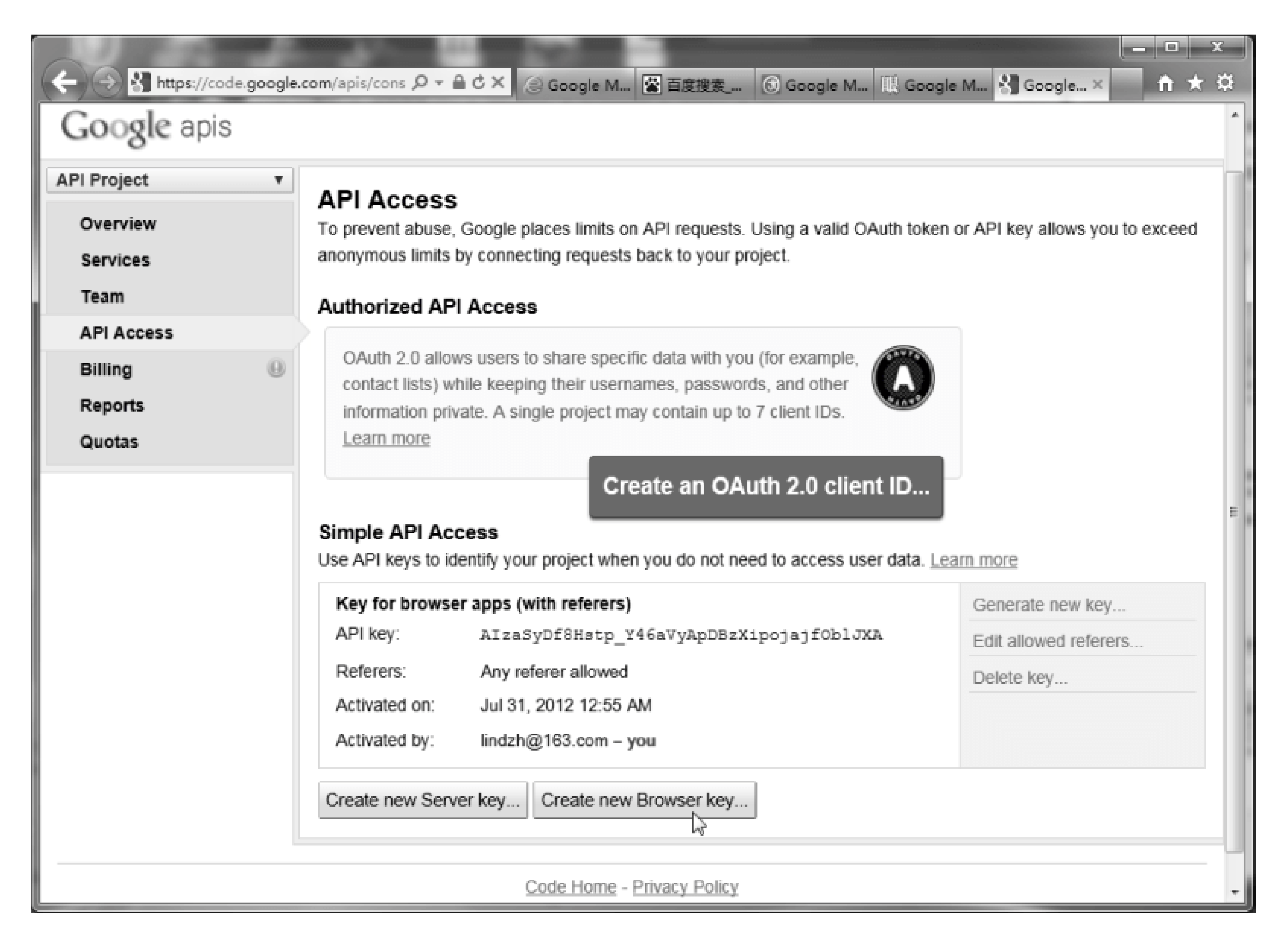


图 10-32　生成 API Key

3）复制 Map API Key

通常在应用 Google Map 的 Activity 中，其布局文件要定义一个“MapView”，并且在该标

签的属性 android:apiKey 中要求有 API key,这也就是获取的 Map API Key。

将获取的地图 API 密钥 Map API Key 复制到 main. xml 布局文件的 MapView 标签中,代码如下。

```
<com.google.android.maps.MapView
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:apiKey = "AizaSyDf8Hstp_Y46aVyApDBzXipojajfOblJXA"
    />
```

注意,如果 apiKey 不正确,模拟器则无法加载地图,只会出现网格效果。

2. 创建支持 Google Map 的模拟器

Google 地图并不是标准 Android 类库,并不是所有的 Android 设备都支持该功能。所以运行 Google 地图应用程序,需要创建一个能支持 Google API 设备的模拟器。也就是说,在模拟器创建期间,选择 SDK 的版本时,在选择目标(Target)选项时要选择 Google APIs(Google Inc.),如图 10-33 所示。

图 10-33 创建模拟器

3. 向 AndroidManifest.xml 文件中添加相关内容

在使用 Google Maps 的应用程序中,需要手动在 AndroidManifest. xml 文件的<application>标签内添加如下的<uses-library>标签。

```
<uses - library android:required = "true" android:name = "com.google.android.maps"></uses -
library>
```

使用 Google Maps 还要求对 Internet 的访问权限,因为需要从 Internet 上读取 Google Maps 数据。所以要在 AndroidManifest. xml 文件的<manifest>标签内添加访问网络权限。

```
<uses - permission android:name = "android.permission.INTERNET" />
```

4. Google Map 应用编程

Android 的 Google Map 应用的关键类是 MapView。MapView 位于 com. google. android.

maps 包下，MapView 是显示一幅带有 Google 地图数据的控件，能捕捉按键事件和触摸手势，以实现地图缩放功能，是 Google 地图 API 的一个缩略版。MapView 通过 Overlay 类控制地图。

使用 MapView 对象调用 getController()方法可以获得一个 MapController 对象。通过 MapController 对象可以对 MapView 对象中的地图进行缩放。常用的方法如表 10-5 所示。

表 10-5　MapController 对象的几个方法及说明

方　法	描　述
zoomIn()	缩小一个等级
zoomOut()	放大一个等级
setZoom(int zoomLevel)	直接设置缩放等级，zoomLevel 取值范围为 1～21
animateTo(GeoPoint point)	平滑移动地图视图到指定点。通过 point 来指定要移动的目标地点，同时还可选在到达目的地后运行一个 Runable 对象，或发送一条 Message
setCenter(GeoPoint point)	移动视图到 point 指定的位置，没有平滑移动的效果

下面通过一个案例来说明 Google Map 的应用。

【案例 10.7】 以交通地图、卫星地图两种方式查询指定经纬度坐标的地理位置，并用箭头图标定位该点，并允许用户缩放地图，移动地图。

【说明】 本例是 Google 地图的应用，事先应创建一个 Target 为 Google APIs 的模拟器设置。

【开发步骤及解析】

(1) 获取一个 Map API Key 值。获取方法前面已经有详细描述。

(2) 创建项目。在 Eclipse 中创建一个名为 MyGoogleMap 的 Android 项目。其应用程序名为 MyGoogleMap，选择 Build Target 时勾选 Google APIs，Platform 为 2.3.1，包名为 cn.com.sgmsc.GMap，Activity 组件名为 MyGoogleMapActivity。

(3) 准备图片资源。在 drawable-mdpi 下存放图标文件 arrow.png。

(4) 准备字符串资源。编写 res/layout 目录下的 strings.xml 文件，代码如下所示。

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <resources>
3      <string name = "hello">Hello World, MyGoogleMap!</string>
4      <string name = "app_name">MyGoogleMap</string>
5      <string name = "txtLong">经度:</string>       <!-- 声明名为 txtLong 的字符串资源 -->
6      <string name = "txtLat">纬度:</string>        <!-- 声明名为 txtLat 的字符串资源 -->
7      <string name = "btnGo">查询</string>          <!-- 声明名为 btnGo 的字符串资源 -->
8      <string name = "etLong">113.32</string>      <!-- 声明名为 etLong 的字符串资源 -->
9      <string name = "etLat">23.136</string>       <!-- 声明名为 etLat 的字符串资源 -->
10     <string name = "satellite">卫星视图</string>  <!-- 声明名为 satellite 的字符串资源 -->
11     <string name = "normal">普通视图</string>     <!-- 声明名为 normal 的字符串资源 -->
12 </resources>
```

第 8、9 行给出程序的初始经度、纬度数据。

(5) 设计布局。编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
```

```
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
    <LinearLayout
     android:orientation = "horizontal"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     >
     <TextView
          android:text = "@string/txtLong"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:layout_gravity = "center_vertical"
          />
     <EditText
          android:id = "@ + id/etLong"
          android:text = "@string/etLong"
          android:layout_width = "110px"
          android:layout_height = "45px"
          />
     <TextView
          android:text = "@string/txtLat"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:layout_gravity = "center_vertical"
          android:paddingLeft = "8px"
          />
     <EditText
          android:id = "@ + id/etLat"
          android:text = "@string/etLat"
          android:layout_width = "110px"
          android:layout_height = "45px"
          />
     </LinearLayout>
<LinearLayout
     android:orientation = "horizontal"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     >
     <RadioGroup
          android:id = "@ + id/rg"
          android:orientation = "horizontal"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:layout_marginRight = "6dip"
          >
          <RadioButton
               android:text = "@string/normal"
               android:id = "@ + id/normal"
               android:checked = "true"
               android:layout_width = "wrap_content"
               android:layout_height = "wrap_content">
               </RadioButton>
          <RadioButton
               android:text = "@string/satellite"
```

```
59                  android:id = "@ + id/satellite"
60                  android:layout_width = "wrap_content"
61                  android:layout_height = "wrap_content">
62                  </RadioButton>
63              </RadioGroup>
64          <Button
65              android:id = "@ + id/btnGo"
66              android:text = "@string/btnGo"
67              android:layout_width = "wrap_content"
68              android:layout_height = "wrap_content"
69              android:layout_marginRight = "6dip"
70              android:layout_weight = "1"
71              />
72          </LinearLayout>
73      <com.google.android.maps.MapView
74              android:id = "@ + id/mv"
75              android:clickable = "true"
76              android:enabled = "true"
77              android:layout_width = "fill_parent"
78              android:layout_height = "fill_parent"
79              android:apiKey = " AizaSyDf8Hstp_Y46aVyApDBzXipojajfOblJXA "
80              />
81      </LinearLayout>
```

① 第 73～80 行定义了该应用的 Google 地图的显示控件 MapView，因为它是外部控件，所以需要在前面加上 com. google. android. maps 包名。

② 第 79 行是 Map API Key 属性，需要输入一串 Google 的 Map API Key 值。只有获取正确的 Map API Key 值才能正常显示 Google 地图。

(6) 开发逻辑代码。打开 src/cn. com. sgmsc. GMap 包下的 MyGoogleMapActivity. java 文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.GMap;

import java.util.List;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;
import android.widget.RadioGroup.OnCheckedChangeListener;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;

public class MyGoogleMapActivity extends MapActivity {    //继承自 MapActivity 的子类
 MapView mv;                                               //声明 MapView 对象引用
```

```
MapController controller;                          //声明 MapController 对象引用
Bitmap bmpArrow;                                   //声明 Bitmap 对象引用
RadioButton rbNormal;                              //声明 RadioButton 对象引用
RadioButton rbSatellite;                           //声明 RadioButton 对象引用
@Override
protected void onCreate(Bundle icicle) {           //重写 onCreate()方法
    super.onCreate(icicle);
    setContentView(R.layout.main);
    bmpArrow = BitmapFactory.decodeResource(getResources(), R.drawable.arrow);
                                                   //获得"箭头"图片
    Button btnGo = (Button)findViewById(R.id.btnGo); //获得 Button 对象
    mv = (MapView)findViewById(R.id.mv);           //获得 MapView 对象
    controller = mv.getController();               //获得 MapController 对象
    mv.setBuiltInZoomControls(true);               //设置是否显示"放大/缩小"按钮
    mv.setSatellite(false);                        //设置不显示卫星模式地图
    mv.setStreetView(false);                       //设置不显示街景模式地图
    mv.setTraffic(true);                           //设置显示交通模式地图

    btnGo.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            EditText etLong = (EditText)findViewById(R.id.etLong); //获得 EditText 对象
            EditText etLat = (EditText)findViewById(R.id.etLat);   //获得 EditText 对象
            String sLong = etLong.getEditableText().toString().trim();  //获得输入的经度
            String sLat = etLat.getEditableText().toString().trim();   //获得输入的纬度
            if(sLong.equals("") || sLat.equals("")){  //判断是否输入空值
                Toast.makeText(MyGoogleMapActivity.this, "对不起,请输入正确的经纬度
                坐标!",
                            Toast.LENGTH_LONG).show();
                return;
            }
            double dLong = Double.parseDouble(sLong);
            double dLat = Double.parseDouble(sLat);
            updateMapView(dLat, dLong);            //调用方法更新 MapView
        }
    });
    btnGo.performClick();
                               //生成一个单击按钮的事件,即相当于单击了一下按钮

    RadioGroup rg = (RadioGroup)findViewById(R.id.rg);//获得 RadioGroup 对象
    rbNormal = (RadioButton)findViewById(R.id.normal);  //获得 RadioButton 对象
    rbSatellite = (RadioButton)findViewById(R.id.satellite);  //获得 RadioButton 对象
    rg.setOnCheckedChangeListener(new OnCheckedChangeListener() {
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            if(checkedId == rbNormal.getId()){     //判断按下的是否是正常视图
                mv.setSatellite(false);
                mv.setStreetView(false);
                mv.setTraffic(true);
            }
            else if(checkedId == rbSatellite.getId()){  //判断按下的是否为卫星视图
                mv.setSatellite(true);
                mv.setStreetView(false);
                mv.setTraffic(false);
            }
        }
    });
}
```

```
78  @Override
79  protected boolean isRouteDisplayed() {                //重写 isRouteDisplayed()方法
80        return false;                                   //不需要在地图上绘制导航线路
81  }
82  //方法:更新 MapView 的视图
83  public void updateMapView(double dLat,double dLong){
84        GeoPoint gp = new GeoPoint((int)(dLat * 1E6), (int)(dLong * 1E6));
85        mv.displayZoomControls(true);                   //设置显示"放大/缩小"按钮
86        controller.animateTo(gp);                       //将地图移动到指定的地理位置
87        List<Overlay> ol = mv.getOverlays();            //获得 MapView 的 Overlay 列表
88        ol.clear();
89        ol.add(new FixOverLay(gp,bmpArrow));            //添加一个新的 Overlay
90  }
91 }
```

① 第 37～39 行,设置地图的显示类型。

② 第 54 行调用 updateMapView(dLat, dLong)方法更新 MapView 中的地图。

③ 第 57 行调用 btnGo. performClick()方法,该方法是由程序控制执行按钮的单击事件,不需要用户操作。

④ 第 62～75 行定义了单选按钮的监听器。在该监听的回调方法内定义了当用户选择哪一种地图的显示方式时,就在 MapView 中显示相应的地图类型。

⑤ 第 83～90 行,定义 updateMapView()方法。在其中第 84 行定义一对经纬度数据,GeoPoint 类位于 com. google. android. maps 包下,以纬度的整数形式存储,表示一对经、纬度值,GeoPoint 对象构造后不可再修改经、纬度值,但可返回该对象。第 88 行删除地图上的原有覆盖。第 89 行添加一个新的 Overlay 覆盖,并调用 FixOverLay 类来标记位置图标,这里 gp 为经、纬度坐标点,bmpArrow 为显示的图标对象。Overlay,即覆盖,可将它理解成一块透明的画布,将它铺在地图上。Overlay 类可以进行的操作主要有两项:绘制(标记)和处理触摸事件。

(7) 开发在地图上标记位置图标的代码。

若要在地图上标记位置,就需要用到 Overlay 类。Overlay 类是一种专门用于在地图上用 2D 图像进行标记的类。由于其默认的绘制方法 draw()并不绘制任何图像,所以通常会开发一个 OverLay 的子类,并添加到 MapView 的 OverLay 列表中。

在 src/cn. com. sgmsc. Gmap 包下创建一个代码文件 FixOverLay. java,该文件是定义一个 OverLay 子类,用于在地图上标记当前位置。其代码如下所示。

```
package cn.com.sgmsc.GMap;                                //声明包语句

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Point;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.Projection;

public class FixOverLay extends Overlay{
 Bitmap bmpArrow;                                         //声明 Bitmap 对象的引用
 GeoPoint gp;                                             //声明 GeoPoint 对象的引用
 public FixOverLay(GeoPoint gp,Bitmap bmp){
      super();                                            //调用父类构造器
      this.gp = gp;                                       //初始化 GeoPoint
```

```
17         bmpArrow = bmp;                                    //初始化 Bitmap
18     }
19     @Override
20     public void draw(Canvas canvas, MapView mapView, boolean shadow) {
21         if(!shadow){
22             Projection proj = mapView.getProjection(); //获得 Projection 对象
23             Point p = new Point();
24             proj.toPixels(gp, p);                      //将真实地理坐标转化为屏幕上的坐标
25             canvas.drawBitmap(bmpArrow,
26                     p.x - bmpArrow.getWidth()/2,
27                     p.y - bmpArrow.getHeight(),
28                     null);                             //绘制箭头图片
29         }
30     }
31 }
```

① 第 20～30 行重写了 draw()方法。该方法有三个参数：参数 canvas，就是所要绘制其上的画布；参数 mapView，所需标记的地图图层，这个参数的一个重要作用就是通过 MapView. getProjection()得到一个 Projection 对象，它完成从物理经纬度到屏幕投影坐标的转换，继而在相应坐标点上进行绘制；参数 shadow，当该值为 true 时，则需绘制阴影图层，若为 false，则只需绘制本来的内容即可。

② 第 25～28 行完成一个图标在指定坐标上的绘制。

(8) 添加标签和权限。打开根目录下的 AndroidManifest. xml，在<manifest>标签内添加下列权限，代码为：

```
<uses-permission android:name="android.permission.INTERNET" />
```

在 AndroidManifest. xml 文件的< application>标签内添加<uses-library>标签：

```
<uses-library android:required="true" android:name="com.google.android.maps"></uses-library>
```

【运行结果】 在 Eclipse 中启动 Google APIs 的 Android 模拟器，然后运行 MyGoogleMap 项目。在界面上显示预设置经纬度的普通视图，如图 10-34 所示，如果单击“卫星视图”单选按钮，则显示如图 10-35 所示的卫星地图。



图 10-34 指向广州市天河城的普通视图



图 10-35 指向广州市天河城的卫星视图

10.5 应用项目签名与打包

如果要将自己开发的 Android 项目发布应用，就必须为自己的应用程序项目打包和签名，将打包签名后的 apk 文件传入 Android 模拟器或 Android 手机中即可安装运行。

10.5.1 Android 应用项目的签名文件

为什么要对应用程序项目进行签名呢？这是因为全世界开发 Android 项目的人很多，在项目中的包和各个类的命名方面，完全有可能大家都使用了相同的名字，这时候如何区分？签名就起着区分的作用。Android 签名是发送者的身份认证，是保证信息传输的完整性的一种手段，是 Android Market 对软件的要求，它可以防止交易中抵赖事情的发生。

1. 关于 Android 签名

在为 Android 应用程序项目签名时，需要注意以下几点。

(1) 所有的 Android 应用都必须有数字签名，没有不存在数字签名的应用，包括模拟器上运行的。Android 系统不会安装没有数字证书的应用。

(2) 同一个开发者的多个应用程序项目尽可能使用同一个数字证书签名，这样做有利于程序升级，有利于程序的模块化设计和开发，可以通过权限(permission)的方式在多个程序间共享数据和代码。

(3) 签名的数字证书不需要权威机构来认证，是开发者自己产生的数字证书，即所谓的自签名。

(4) 正式发布一个 Android 应用时，必须使用一个合适的私钥生成的数字证书来给程序签名，不能使用 ADT 插件或者 ANT 工具生成的调试证书来发布。

(5) 数字证书都是有有效期的，Android 只是在应用程序项目安装的时候才会检查证书的有效期。如果程序已经安装在系统中，即使证书过期也不会影响程序的正常功能。

2. 生成签名文件

在 Android 中生成签名文件有两种方式：一是使用命令行状态下的命令方式，二是使用 ADT 插件方式。注意：在生成签名文件时，保存签名文件的文件夹必须事先建立，生成签名文件过程中不会自动创建文件夹。

1) 使用命令行方式

如果在 Windows 操作系统的系统环境变量 path 中加入“C:\Program Files\Java\jdk1.7.0_05\bin”路径，那么使用 cmd 命令进入命令行状态后，先进入保存签名文件的文件夹中(使用 CD ＜文件夹名＞)，再输入命令(如图 10-36 所示)：

```
Keytool -genkey -alias lindand.keystore -keyalg RSA -validity 20000 -keystore lindand.keystore
```



图 10-36 在命令行状态下输入生成签名文件命令

在输入完上述命令,按回车键之后,系统会给出一系列的问题。根据屏幕上的提示逐一回答,形成签名文件,签名文件的扩展名为.keystore,如图10-37所示。

```
C:\Windows\system32\cmd.exe
E:\mypublic>keytool -genkey -alias lindand.keystore -keyalg RSA -validity 20000
-keystore lindand.keystore
输入keystore密码:
再次输入新密码:
您的名字与姓氏是什么?
  [Unknown]:   lindazh
您的组织单位名称是什么?
  [Unknown]:   sgmsc.com.cn
您的组织名称是什么?
  [Unknown]:   3gms
您所在的城市或区域名称是什么?
  [Unknown]:   GuangZhou
您所在的州或省份名称是什么?
  [Unknown]:   GuangDong
该单位的两字母国家代码是什么
  [Unknown]:   cn
CN=lindazh, OU=sgmsc.com.cn, O=3gms, L=GuangZhou, ST=GuangDong, C=cn 正确吗?
  [否]:  Y

输入<lindand.keystore>的主密码
        (如果和 keystore 密码相同, 按回车):
再次输入新密码:

E:\mypublic>_
```

图10-37 在命令行状态下输入生成签名文件命令

其中,lindand.keystore为签名文件的文件名,E:\中的mypublic为保存lindand.keystore的文件夹;-validity 20000为证书的有效天数。另外,在输入密码时是不能回显的,所以输入时既看不到任何光标移动,也看不到输入了符号,这个密码在给.apk文件签名的时候需要。

2) 使用ADT插件方式

首先在Eclipse中选择需要发布的项目,然后右击,在弹出菜单中依次选择Android Tools→Export Signed Application Packge...,接下来按照打包并生成签名文件向导的提示逐一完成生成签名文件。具体操作步骤将在10.5.2节中详细介绍。

10.5.2 Android应用项目的打包

1. ADT自动生成的apk

在Eclipse中创建的Android应用项目要运行在手机上,必须要先打包再安装。Android应用项目打包即为生成apk文件,apk(Android Package)是Android的安装包,它包含应用项目的二进制代码、资源、配置文件等。每个要安装到Android平台的应用项目都要被编译打包为一个单独的.apk文件。实际上,在Eclipse中创建应用项目并运行的过程就是:编译→打包→安装→运行。经过编译之后在项目的bin目录下会自动生成一个apk文件。与J2ME不同,无须手动打包,而且如果代码有改动就自动build,生成新的apk文件。

这个过程是通过Eclipse的ADT插件自动完成的,并不适合发布应用项目。Android应用程序打包必须要通过一个证书文件进行加密,而在调试时之所以也能打包,是因为Eclipse在安装ADT插件的时候创建了一个keystore证书文件。它的位置可以通过依次选择Eclipse的菜单Window→Preferences→Android→Build下面找到,如10.4.2节中的图10-27所示。

2. 发布应用使用的apk

如果要发布Android应用项目,打包必须要通过一个证书文件进行加密。这样的打包文

件生成步骤如下。

(1) 在 Eclipse 中,选择需要发布的项目,然后右击,弹出菜单;在弹出菜单中选择 Android Tools→Export Signed Application Package...,该选项是完成打包并同时生成签名文件操作,或选择 Android Tools→Export UnSigned Application Package...,该选项是完成打包但不生成签名文件操作,如图 10-38 所示。

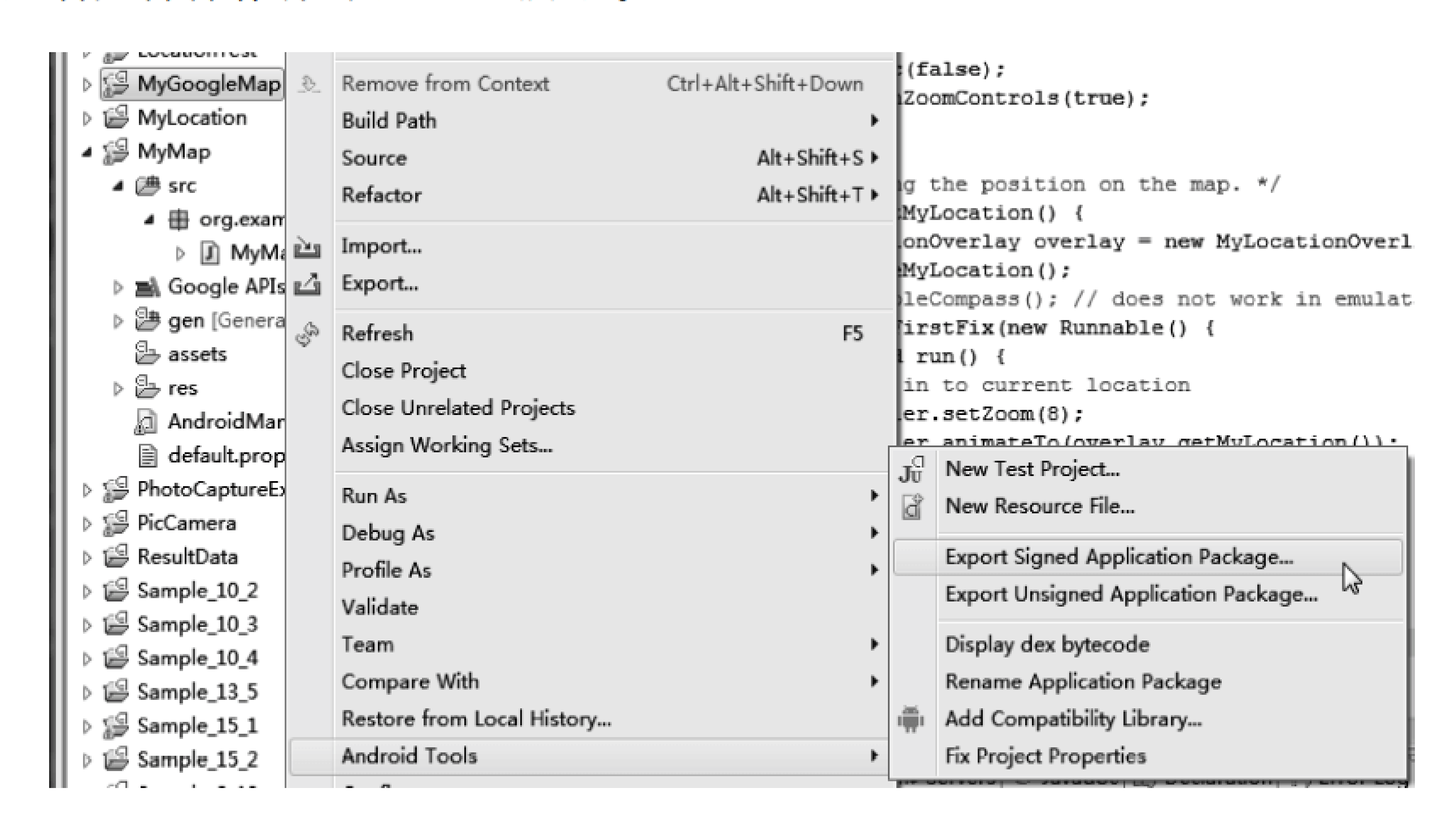


图 10-38　选择打包菜单项

这里以选择 Export Signed Application Package...为例,说明打包并生成签名文件的过程。单击 Export Signed Application Package...项,进入 Export Android Application(输出 Android 应用)向导页,如图 10-39 所示。

(2) 单击 Next 按钮进入签名文件的定义页。在此页,选择 Create new keystore 项表示是新创建签名文件,定义签名文件的文件名、所在路径、密码等信息,如图 10-40 所示。

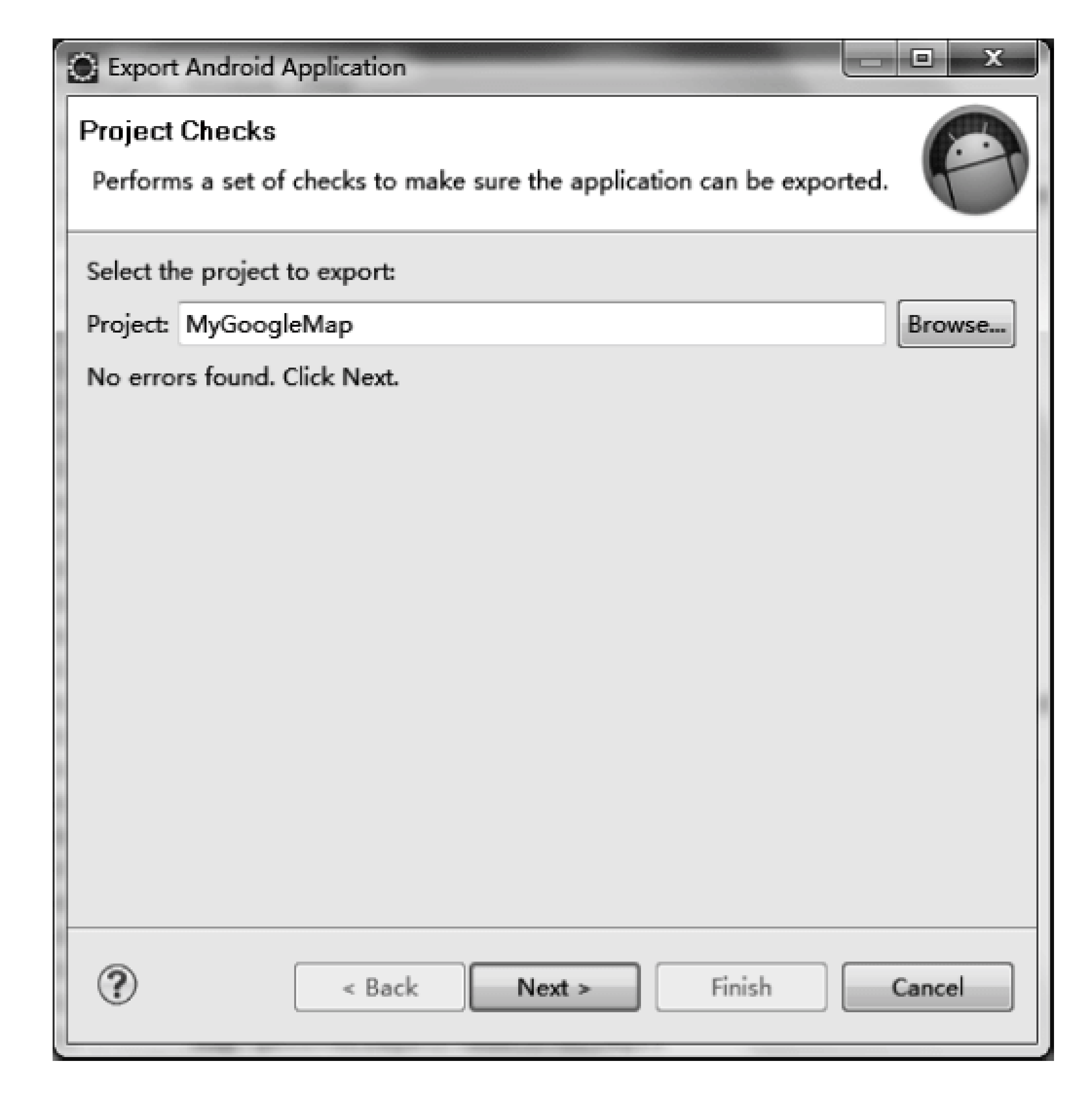



图 10-39　Export Android Application 的选择项目页

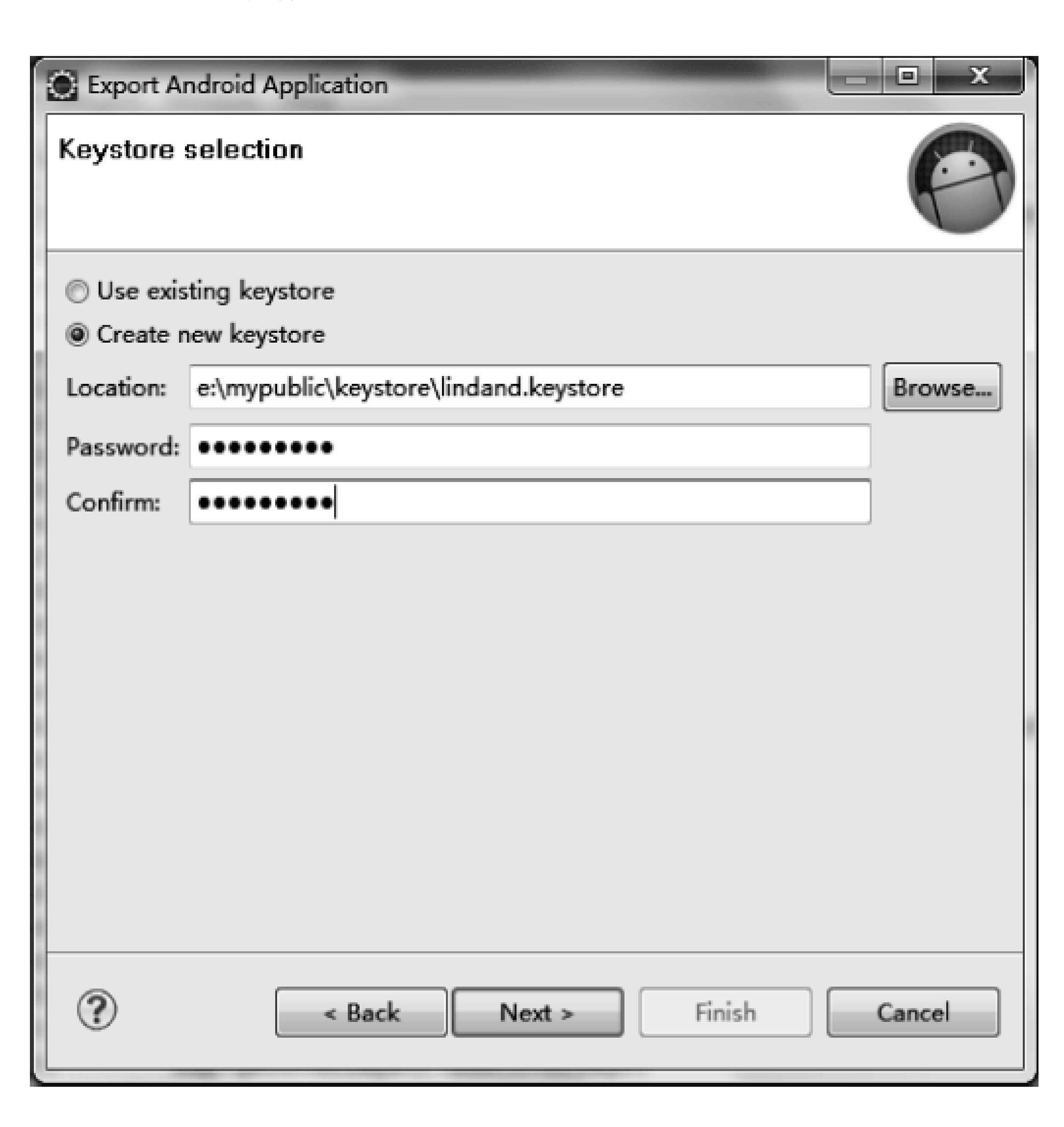


图 10-40　创建签名文件的定义页

(3) 单击 Next 按钮进入生成签名文件信息页。在此页,输入签名文件名、密码、有效时间(以年为单位),以及签名者的相关信息等,如图 10-41 所示。

（4）单击 Next 按钮进入生成打包文件定义页。在此页，定义打包文件的文件名，保存路径。然后单击 Finish 按钮，打包完毕，如图 10-42 所示。



图 10-41 生成签名文件信息页



图 10-42 生成打包文件页

10.5.3 Android 应用项目的打包签名

在签名文件生成后，紧接着是要对应用项目的 apk 文件进行签名。为未签名的 apk 包签名也有两种方式：一是使用命令行方式，二是使用 ADT 插件方式。

1. 使用命令行方式

进入命令行状态，在存放签名文件的文件夹中输入命令：

```
jarsigner -verbose -keystore keystore/lindand.keystore -signedjar mygooglemap_s.apk
mygooglemap.apk lindand.keystore
```

在输入完命令之后回车，再输入密码（无回显），接下来系统将进行签名，如图 10-43 所示。

```
E:\mypublic>jarsigner -verbose -keystore keystore/lindand.keystore -signedjar my
googlemap_s.apk mygooglemap.apk lindand.keystore
输入密钥库的口令短语：
   正在添加： META-INF/LINDAND_.SF
   正在添加： META-INF/LINDAND_.RSA
  正在签名： res/layout/main.xml
  正在签名： AndroidManifest.xml
  正在签名： resources.arsc
  正在签名： res/drawable-hdpi/icon.png
  正在签名： res/drawable-ldpi/icon.png
  正在签名： res/drawable-mdpi/arrow.png
  正在签名： res/drawable-mdpi/icon.png
  正在签名： classes.dex

E:\mypublic>
```

图 10-43 在 DOS 命令行状态下完成签名

命令中，mygooglemap_s.apk 是签名后的包文件，mygooglemap.apk 为未签名的包文件。完成签名后，为了确认是否签名成功，可以输入下列命令给予验证：

jarsigner -verify mygooglemap_s.apk

如果出现“jar 已验证”，说明签名成功，如图 10-44 所示。

```
E:\mypublic>jarsigner -verify mygooglemap_s.apk
jar 已验证。

E:\mypublic>
```

图 10-44　验证签名是否成功

2. 使用 ADT 插件方式

仍是使用 Export Signed Application Package...选项。其操作步骤如下。

(1) 选择需要发布的项目，然后右击。

(2) 在弹出菜单中选择 Android Tools→Export Signed Application Package...。

(3) 在 Keystore selection 页(如图 10-40 所示)中，选择 Use existing keystore，输入密码，然后单击 Next 按钮，验证签名成功。

小结

本章主要介绍了 Android 平台下进行网络数据通信、网页浏览及地理位置信息应用等与网络有关的知识和应用。通过案例展示了 Android 的几种常见网络应用技术。在使用 Socket 开发时要注意通信过程的顺序；使用 URL 和 HTTP 获取网络服务器上信息时，需要正确地搭建 Tomcat 服务器，要掌握 GET 和 POST 上传数据的区别用法，这些都是非常有用的网络编程技术。在地理位置定位中，要掌握好 LocationManager 类的常用方法的用法；在使用 Google Map 时要会申请 Map API Key。学会了这些技术，就可以开发面向网络的应用了。

至此，读者已经学习了大量的 Android 编程技术，具备了开发一般应用项目的能力，可以将自己开发的项目发布应用了。所以在本章的最后一节介绍了应用程序项目签名打包的操作方法。在前面各章节的介绍中，涉及手机功能的应用还很少，但 Andriod 系统毕竟是始于手机的操作系统，与手机相关的应用开发是不可或缺的，第 11 章将补上这一缺失。

练习

1. 使用 Tomcat 服务器(端口号为 8080)，运用线程实现多客户端的连接通信功能。要求服务器端在控制台、客户端在运行界面都能显示登录服务器的 IP 地址和登录的次数这些信息。运行结果可参考图 10-45 和图 10-46。

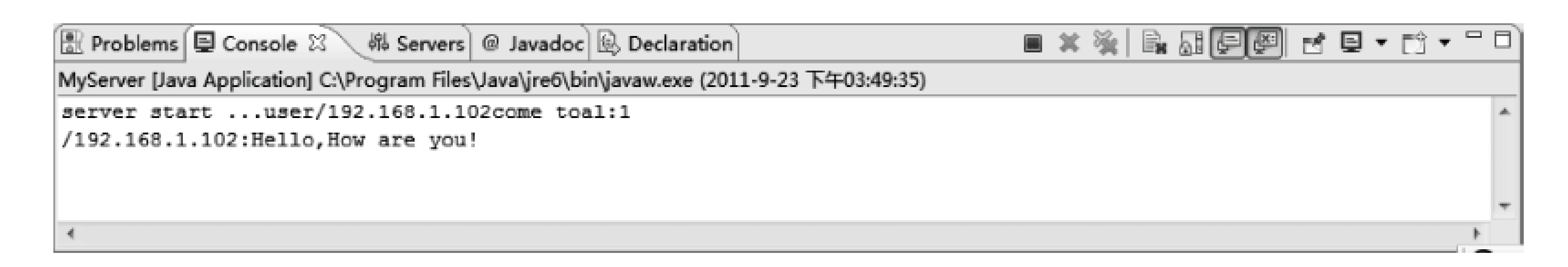


图 10-45　服务器端在控制台的显示结果



图 10-46 客户端在屏幕上的显示结果

2. 完善“掌上博客”：将登录信息上传至服务器中进行验证。

3. 设计一个可以向前、向后浏览网页的应用。

4. 设计一个以交通地图的形式，即时显示你的地理位置的应用，即一个简单的 GPS 定位，并签名打包。

第11章 手机基本功能开发

本章将对手机的一些特有功能的开发进行介绍，包括收发短信、E-mail 邮件，拨打电话，查看手机的设置和状态，以及传感器等。

11.1 短信控制

短信服务(Short Message Service，SMS)在手机使用中几率比较大。尽管手机中都有定制短信管理工具，但有时候用户需要按自己的意图来定义短信收发管理，所以掌握管理短信的相关功能开发在手机应用中非常有用。对手机的短信控制可包括短信发送、接收和群发短信，以及查询发送的状态等。

11.1.1 发送短信

Android 发送短信其实很简单，关键的类是 SmsManager，它位于 android. telephony 包下。SmsManager 类是发送短信的工具类。每个 SmsMessage 对象都包含短信的详细信息，例如：发送者的电话号码，时间戳和短信息内容。SmsMassage 字符串的表现形式是一组 pdus 串。SmsManager 类常用于发送短信的方法如表 11-1 所示。

表 11-1 SmsManager 类常用的方法及说明

方　法	描　述
divideMessage(String text)	当短信超过 SMS 消息的最大长度时，将短信分割为几段。其中，text 是初始的消息，不能为空。返回值为有序的 ArrayList<String>，可以重新组合为初始的消息
getDefault()	获取 SmsManager 的默认实例。返回值为 SmsManager 的默认实例
sendTextMessage (String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)	发送一个基于 SMS 的文本。其中，destination Address 是消息的目标地址；scAddress 是服务中心地址；text 是初始的消息；sentIntent 是当消息成功发送或失败，这个对象就广播，这个参数最好不为空，否则会存在资源浪费的潜在问题；deliveryIntent 是当消息成功发送到接收者这个对象就广播
sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList < String > parts, ArrayList <PendingIntent> sentIntents, ArrayList <PendingIntent> deliveryIntents)	发送一个基于 SMS 多部分文本，调用者应用已经通过调用 divideMessage(String text)将消息分割成正确的大小。参数与 sendTextMessage 方法中一样，不过 sentIntents 和 deliveryIntents 是一组 PendingIntent

注意，开发发送短信的应用时，要在 AndroidManifest.xml 中添加发送 SMS 权限，否则应用程序无法执行发送短信任务。具体做法是在<manifest>标签下添加如下语句：

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

下面通过一个发送短信的案例来说明其用法。

【案例 11.1】 设计一个可发送短信的应用，要求当短信的字数超过 70（以 Unicode 字符单位计）时进行分割。

【说明】 本例只开发发送短信的应用，接收短信的操作由系统的短信管理程序负责处理。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 SMSSendDemo 的 Android 项目。其应用程序名为 SMSSendDemo，包名为 cn.com.sgmsc.sendSMS，Activity 组件名为 SMSSendDemoActivity。

(2) 准备字符串资源。编写 res/layout 目录下的 strings.xml 文件，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="hello">Hello World, SMSSendDemoActivity!</string>
    <string name="app_name">SMSSendDemo</string>
    <string name="send">发送短信</string>
    <string name="sms">短信内容</string>
    <string name="tel">对方号码</string>

</resources>
```

(3) 设计布局。编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="top"><!-- 添加一个垂直的线性布局 -->
    <TextView
        android:text="@string/tel"
        android:id="@+id/tv01"
        android:textSize="20dip"
        android:textStyle="bold"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="5dip"/><!-- 添加一个 TextView 控件 -->
    <EditText
        android:id="@+id/smstel"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textStyle="normal"
        />
        <!-- 添加一个 EditText 控件 -->
    <TextView
        android:text="@string/sms"
        android:id="@+id/tv02"
        android:layout_width="wrap_content"
        android:textSize="20dip"
```

```
        android:textStyle = "bold"
        android:paddingLeft = "5dip"
        android:layout_height = "wrap_content"/><!-- 添加一个 TextView 控件 -->
    <EditText
        android:id = "@ + id/smsContent"
        android:layout_width = "fill_parent"
        android:singleLine = "false"
        android:gravity = "top|left"
        android:layout_height = "280dip"
        android:textStyle = "normal"
        />
        <!--  添加一个 EditText 控件 -->
    <Button
        android:text = "@string/send"
        android:id = "@ + id/smsSend"
        android:textSize = "20dip"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"/><!--  添加一个 Button 控件 -->
</LinearLayout>
```

(4) 开发逻辑代码。打开 src/cn. com. sgmsc. sendSMS 包下的 SendSMSDemoActivity. java 文件,并编辑之。其代码如下所示。

```
package cn.com.sgmsc.sendSMS;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.util.List;

public class SMSSendDemoActivity extends Activity {

    private EditText txtNo;
    private EditText txtContent;
    private Button btnSend;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        txtNo = (EditText) findViewById(R.id.smstel);
        txtContent = (EditText) findViewById(R.id.smsContent);
        btnSend = (Button) findViewById(R.id.smsSend);

        btnSend.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                String strNo = txtNo.getText().toString();
```

```
33                String strContent = txtContent.getText().toString();
34                SmsManager smsManager = SmsManager.getDefault();
35                PendingIntent sentIntent =
36                        PendingIntent.getBroadcast(SMSSendDemoActivity.this, 0, new Intent(), 0);
37                //如果字数超过 70(个 Unicode 字符),需拆分成多条短信发送
38                if (strContent.length() > 70) {
39                    //使用短信管理器进行短信内容的分段,返回分成的段
40                    List<String> msgs = smsManager.divideMessage(strContent);
41                    for (String msg : msgs) {
42                        //使用短信管理器发送短信内容
43                        smsManager.sendTextMessage(strNo, null, msg, sentIntent, null);
44                    }
45                } else {
46                    smsManager.sendTextMessage(strNo, null, strContent, sentIntent, null);
47                }
48                Toast.makeText(SMSSendDemoActivity.this, "短信发送完成", Toast.LENGTH_LONG).
                  show();
49            }
50        });
51    }
52 }
```

① 第 34 行,创建一个默认的 SmsManager 对象。

② 第 35 行,创建一个 PendingIntent 对象,作为发送短信时的一个广播对象。

③ 第 40 行,调用 divideMessage()方法,将超出 70 个字的短信分割成多条短信,存放在 List<String>对象中。

④ 第 41～44 行,如果有分割多条短信,通过一个循环,对 List<String>对象中的每一条短信,调用 sendTextMessage()方法进行发送。

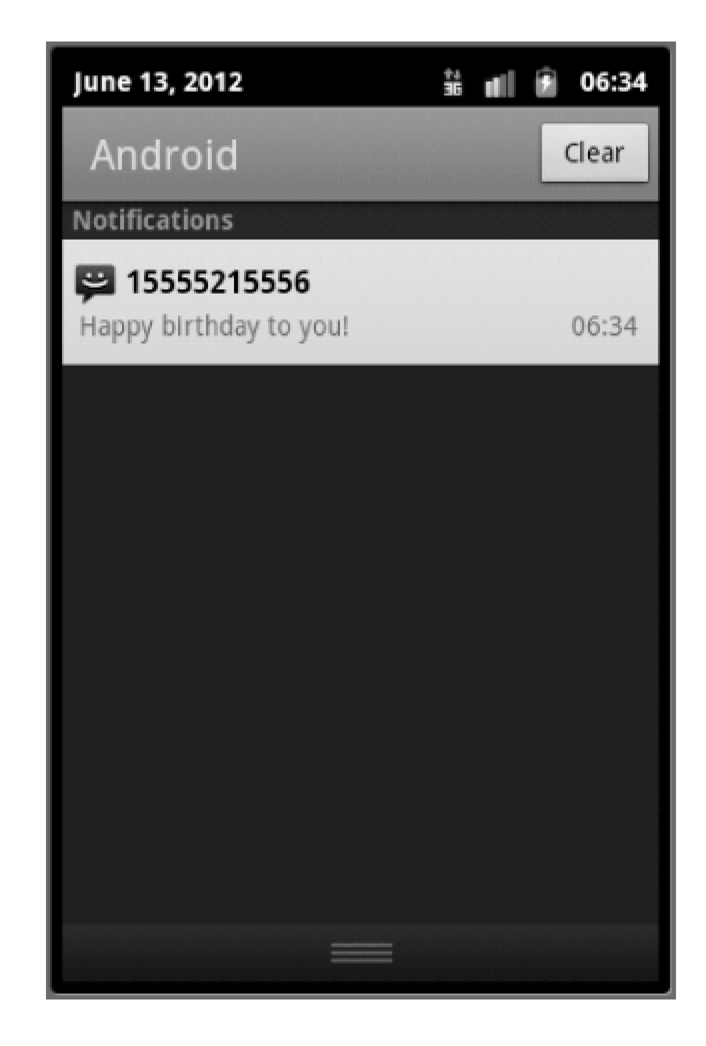
(5) 添加权限。打开根目录下的 AndroidManifest.xml 文件,在<manifest>标签内添加下列权限:

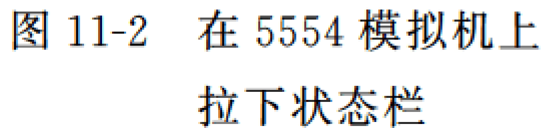
```
<uses-permission android:name="android.permission.SEND_SMS"/><!-- 添加权限 -->
```

【运行结果】 在 Eclipse 中启动两个 Android 模拟器,其模拟器号分别为 5554 和 5556。然后运行 SMSSendDemo 项目。如果系统选用 5556 模拟器运行程序,在其屏幕上可输入对方的手机号,在本例中使用的是 5554;再输入短信内容,然后单击“发送短信”按钮,将短信发送到指定方的手机上,如图 11-1 所示。这时将从 5554 模拟器的状态栏中看到接收到短信的通知。向下拉开状态栏,可以看到收到一条来自 15555215556 的短信,如图 11-2 所示;双击该通知可以打开看到详细的短信内容,如图 11-3 所示。当然,也可不下拉开状态栏,而是单击 menu 按钮,在屏幕上出现许多应用程序,其中有个 Messaging 应用,如图 11-4 所示,它就是系统提供的短信管理程序。打开它也可以看到详细的短信内容。



图 11-1 从 5556 模拟机上发出短信




图 11-2　在 5554 模拟机上拉下状态栏

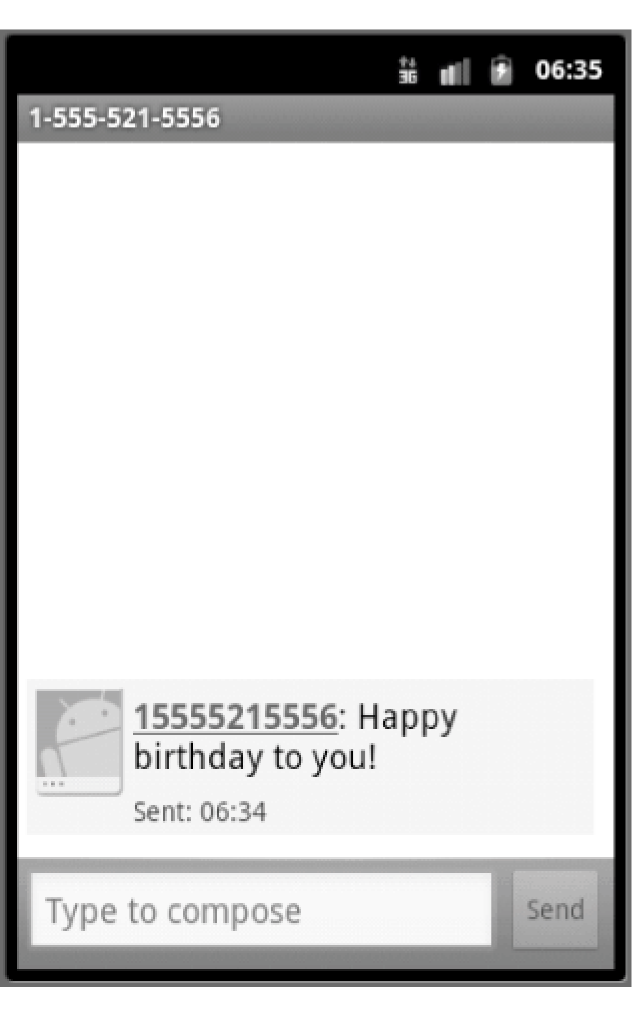


图 11-3　浏览详细短信内容

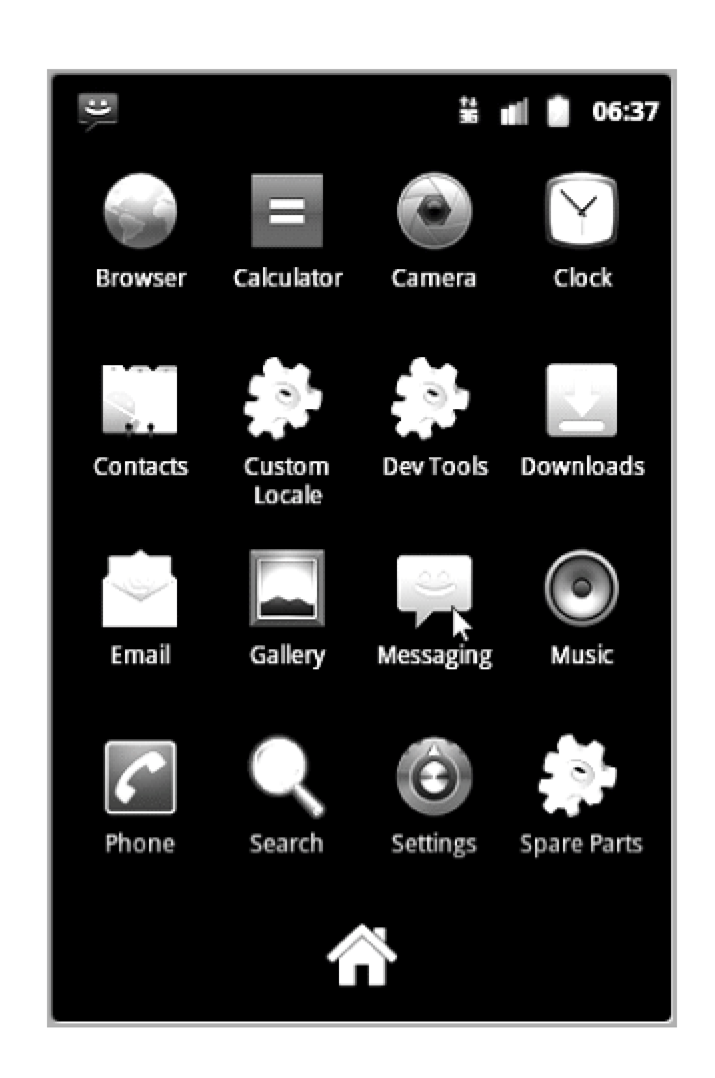


图 11-4　通过 menu 键调出应用程序图标

11.1.2　群发短信

群发短消息，实际上是设置一个循环，对需要发送的号码逐一发送即可。把需要发送的联系人信息记录在一个 HashMap 中，HashMap 是一种非常常用的数据类型——“链表散列”。群发时，通常都是从手机通讯录中选取联系人，然后存放到 HashMap 中。其编程过程及关键的代码如下。

1. 切换到通讯录

```
Uri uri = Uri.parse("content://contacts/people");
Intent intent = new Intent(Intent.ACTION_PICK, uri);   //创建 Intent
startActivityForResult(intent, 1);                     //切换到通讯录
```

2. 从通讯录中得到联系人姓名和电话号码

```
ContentResolver cr = getContentResolver();             //得到 ContentResolver 对象
Cursor c = managedQuery(myUrl, null, null, null, null);
c.moveToFirst();
int nameFieldColumnIndex = c.getColumnIndex(PhoneLookup.DISPLAY_NAME);//取得联系人名字
String sName = c.getString(nameFieldColumnIndex);                    //得到姓名
String contactId = c.getString(c.getColumnIndex(ContactsContract.Contacts._ID));
                                                                     //取得联系人 ID
Cursor phone = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
ContactsContract. CommonDataKinds.Phone.CONTACT_ID + " = " + contactId,null, null);
String strPhoneNumber = "";            //取得电话号码(如果存在多个号码,只取一个)
If(phone.moveToNext()) {               //得到一个电话号码
    strPhoneNumber =
      phone.getString(phone.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
}
Peoples.put(sName, strPhoneNumber);    //存放到容器中
```

3. 取 HashMap 中的值

Iterator 是对集合进行迭代的迭代器，可实现对集合元素的遍历。在这里，使用 Iterator 取 HashMap 中的集合元素值。

```
HashMap<String, String> peoples = new HashMap<String, String>();  //存储所有选择的
                                                                  //联系人信息
…
Set keySet = peoples.keySet();             //得到键值集合
Iterator ii = keySet.iterator();
people.setText("");                        //置空
while(ii.hasNext()){
     Object key = ii.next();               //得到键值
     String tempName = (String)key;        //姓名
     String tempPhone = peoples.get(key);  //得到电话号码
     …;
}
```

11.1.3 接收短信

根据接收短信时系统广播的 android.provider.Telephony.SMS_RECEIVED 的 Intent，可以扩展一个 BroadcastReceiver 来接收 SMS。BroadcastReceiver 是在后台进行监听，一旦接收到短信，就会触发运行。使用 BroadcastReceiver 需要注册，如果在 AndroidManifest.xml 中注册，需要在<application>标签下加入下列代码：

```
<receiver android:name=".MyBroadcastReceiver">
  <intent-filter>
    <action  android:name="android.provider.Telephony.SMS_RECEIVED"/>
  </intent-filter>
</receiver>
```

并且还要在 AndroidManifest.xml 的<manifest>下添加接收短信的权限。具体语句如下：

```
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
```

另外，在接收短信的应用中需要编写一个 SMS 的广播接收器类，在该类中要有以下 4 个关键性代码。

(1) 接收短信的 Broadcast Action 是 android.provider.Telephony.SMS_RECEIVED，因此，要在 onReceiver()方法的开始部分判断接收到的是否是接收短信的 Broadcast Action。

(2) 需要通过 Bundle.get("pdus")来获得接收到的短信消息。这个方法返回了一个表示短信内容的数组。每一个数组元素表示一条短信。这就意味着通过 Bundle.get("pdus")可以返回多条系统接收到的短信内容。

(3) 通过 Bundle.get("pdus")返回的数组一般不能直接使用，需要使用 SmsMessage.createFromPdu()方法将这些数组元素转换成 SmsMessage 对象才可以使用。每一个 SmsMessage 对象表示一条短信。

(4) 通过 SmsMessage 类的 getDisplayOriginatingAddress()方法可以获得发送短信的电话号码。通过 getDisplayMessageBody()方法可以获得短信的内容。

这里，Android 设备收发 SMS 是以 PDU(Protocol Description Unit)编码形式来传输的。

11.1.4　查询发送状态

当发送短信的命令发出之后，会有多种状态，如发送成功，或发送暂停，或发送失败等。这些状态有时需要反馈给用户。查询短信发送状态需要注册 BroadcastReceiver，并且其查询状态代码要在 BroadcastReceiver 内的 onReceive()方法中编写，根据得到的结果值判断短信发送的状态。

BroadcastReceiver 接收到的 Intent 中包含的状态值如下。

✍Activity. RESULT_OK——发送成功。

✍SmsManager. RESULT_ERROR_GENERIC_FAILURE——发送失败，值为 1。

✍SmsManager. RESULT_ERROR_RADIO_OFF——无线连接异常，值为 2。

✍SmsManager. RESULT_ERROR_NULL_PDU——PDU 为空值，值为 3。

✍SmsManager. RESULT_ERROR_NO_SERVICE——服务不可用，值为 4。

下面通过一个收发短信的案例来说明短信控制应用。

【案例 11.2】　开发一个短信收发的应用，要求发出短信后能提示短信发送的状态信息，对方收到短信后回复，能给出接收回复的简要信息。

【说明】　接收短信的应用是由 Android 系统提供的短信管理应用程序负责。本例是要针对发送短信方的用户体验而编程的，其内容包括发送短信，反馈发送短信状态，接收对方回复短信时的提示信息。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 SMSDemo 的 Android 项目。其应用程序名为 SMSDemo，包名为 cn. com. sgmsc. SMS，Activity 组件名为 SMSSendActivity。

(2) 准备字符串资源。编写 res/layout 目录下的 strings. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>

    <string name = "hello">Hello World, SMSSendActivity!</string>
    <string name = "app_name">SMSDemo</string>
    <string name = "totel">Please input the TEL No:</string>
    <string name = "smscontent">Please input the SMS content:</string>
    <string name = "sendbtn">SEND</string>
    <string name = "message">send successfully!</string>

</resources>
```

(3) 设计布局。编写 res/layout 目录下的 main. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "top"><!-- 添加一个垂直的线性布局 -->
    <TextView
        android:text = "@string/totel"
        android:id = "@ + id/tv01"
```

```
10          android:textSize = "18sp"
11          android:textStyle = "bold"
12          android:layout_width = "wrap_content"
13          android:layout_height = "wrap_content"
14          android:paddingLeft = "5dip"/><! -- 添加一个 TextView 控件 -->
15      <EditText
16          android:id = "@ + id/smsTel"
17          android:layout_width = "fill_parent"
18          android:layout_height = "wrap_content"
19          android:textStyle = "normal"
20          />
21          <! --   添加一个 EditText 控件 -->
22      <TextView
23          android:text = "@string/smscontent"
24          android:id = "@ + id/tv02"
25          android:layout_width = "wrap_content"
26          android:textSize = "18sp"
27          android:textStyle = "bold"
28          android:paddingLeft = "5dip"
29          android:layout_height = "wrap_content"/><! -- 添加一个 TextView 控件 -->
30      <EditText
31          android:id = "@ + id/smsContent"
32          android:layout_width = "fill_parent"
33          android:singleLine = "false"
34          android:gravity = "top|left"
35          android:layout_height = "280dip"
36          android:textStyle = "normal"
37          />
38          <! --   添加一个 EditText 控件 -->
39      <Button
40          android:text = "@string/sendbtn"
41          android:id = "@ + id/smsSend"
42          android:textSize = "20dip"
43          android:layout_width = "fill_parent"
44          android:layout_height = "wrap_content"/><! --   添加一个 Button 控件 -->
45  </LinearLayout>
```

第 34 行，设置该 EditText 居上、居左对齐。当 android:gravity 需要设置多个值时，可以使用“|”进行分隔。

(4) 开发逻辑代码。打开 src/cn.com.sgmsc.SMS 包下的 SendSMSActivity.java 文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.SMS;

import java.util.List;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class SMSSendActivity extends Activity {

String SENT_SMS_ACTION = "SENT_SMS_ACTION";
String DELIVERED_SMS_ACTION = "DELIVERED_SMS_ACTION";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btn = (Button)findViewById(R.id.smsSend);
        btn.setOnClickListener(new View.OnClickListener() {

            public void onClick(View arg0) {
                EditText telNoText = (EditText)findViewById(R.id.smsTel);
                EditText contentText = (EditText)findViewById(R.id.smsContent);

                String telNo = telNoText.getText().toString();
                String content = contentText.getText().toString();
                if (validate(telNo, content))
                {
                    sendSMS(telNo, content);
                }
            }
        });
    }

    /* 发送 SMS 方法 */
    private void sendSMS(String phoneNumber, String message) {

        /* 生成 sentIntent 参数 */
        Intent sentIntent = new Intent(SENT_SMS_ACTION);
        PendingIntent sentPI = PendingIntent.getBroadcast(this, 0, sentIntent, 0);
        /* 生成 deilverIntent 参数 */
        Intent deliverIntent = new Intent(DELIVERED_SMS_ACTION);
        PendingIntent deliverPI = PendingIntent.getBroadcast(this, 0, deliverIntent, 0);

        SmsManager sms = SmsManager.getDefault();
        if (message.length() > 70) {
            List<String> msgs = sms.divideMessage(message);
            for (String msg : msgs) {
                sms.sendTextMessage(phoneNumber, null, msg, sentPI, deliverPI);
            }
        } else {
            sms.sendTextMessage(phoneNumber, null, message, sentPI, deliverPI);
        }
        Toast.makeText(SMSSendActivity.this, R.string.message, Toast.LENGTH_LONG).show();

        /* 注册 BroadcastReceivers,广播短信发送状态 */
        registerReceiver(new BroadcastReceiver(){
            @Override
```

```
                public void onReceive(Context _context, Intent _intent)
                {
                    switch(getResultCode()){
                        case Activity.RESULT_OK:
                            Toast.makeText(getBaseContext(),
                                    "SMS sent success actions",
                                    Toast.LENGTH_SHORT).show();
                            break;
                        case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                            Toast.makeText(getBaseContext(),
                                    "SMS generic failure actions",
                                    Toast.LENGTH_SHORT).show();
                            break;
                        case SmsManager.RESULT_ERROR_RADIO_OFF:
                            Toast.makeText(getBaseContext(),
                                    "SMS radio off failure actions",
                                    Toast.LENGTH_SHORT).show();
                            break;
                        case SmsManager.RESULT_ERROR_NULL_PDU:
                            Toast.makeText(getBaseContext(),
                                    "SMS null PDU failure actions",
                                    Toast.LENGTH_SHORT).show();
                            break;
                    }
                }
            },
            new IntentFilter(SENT_SMS_ACTION));
            registerReceiver(new BroadcastReceiver(){
                @Override
                public void onReceive(Context _context, Intent _intent)
                {
                    Toast.makeText(getBaseContext(),
                            "SMS delivered actions",
                        Toast.LENGTH_SHORT).show();
                }
            },
            new IntentFilter(DELIVERED_SMS_ACTION));
        }
        /* 判断输入信息的有效性 */
        public boolean validate(String telNo, String content){

            if((null == telNo)||("".equals(telNo.trim()))){
                Toast.makeText(this, "please input the telephone No.!",Toast.LENGTH_LONG).show();
                return false;
            }
            if((null == content)||("".equals(content.trim()))){
                Toast.makeText(this, "please input the message content!",Toast.LENGTH_LONG).show();
                return false;
            }
            return true;
        }

}
```

① 第 46～106 行，定义了 sendSMS()方法。其中第 49～64 行是发送短信的代码；第 67～94 行是注册一个 BroadcastReceiver，并在重写其 onReceive()方法中，根据不同的发送状态结果提示不同的消息。第 95 行创建过滤器。

② 第 108～119 行，定义判断输入的手机号码和短信内容是否为空，以此来保证信息的有效性。当然，也可以在此处定义更多的如验证手机号码是否合法等判断代码。

(5) 开发接收短信广播接收器类代码。在 src/cn.com.sgmsc.SMS 包下新建一个名为 SMSReceiver.java 的文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.SMS;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;

public class SMSReceiver extends BroadcastReceiver {

 private static final String strRes = "android.provider.Telephony.SMS_RECEIVED";//收到的
                                                                                 //是短信

 @Override
 public void onReceive(Context arg0, Intent arg1) {
        /* 判断接收到的广播是否为收到短信的 Broadcast Action */
        if(strRes.equals(arg1.getAction())){
            StringBuilder sb = new StringBuilder();
            /* 接收由 SMS 传过来的数据 */
            Bundle bundle = arg1.getExtras();
            /* 判断是否有数据 */
            if(bundle!= null){
                /* 通过 pdus 可以获得接收到的所有短信消息 */
                Object[] pdusArray = (Object[])bundle.get("pdus");
                /* 构建短信对象 array,并依据收到的对象长度来创建 array 的大小 */
                SmsMessage[] msg = new SmsMessage[pdusArray.length];
                for(int i = 0 ;i<pdusArray.length;i++){
                    msg[i] = SmsMessage.createFromPdu((byte[])pdusArray[i]);
                }
                /* 将送来的短信合并自定义信息于 StringBuilder 当中 */
                for(SmsMessage curMsg:msg){
                    sb.append("You got the message From:【");
                    sb.append(curMsg.getDisplayOriginatingAddress()); //获得接收短信的
                                                                      //电话号码
                    sb.append("】Content: ");
                    sb.append(curMsg.getDisplayMessageBody());        //获得短信的内容
                }
                Toast.makeText(arg0,
                          "Got The Message:" + sb.toString(),
                       Toast.LENGTH_SHORT).show();
            }
        }
    }
```

```
43
44 }
```

第 26 行，如果回复的短信过长，会被分割成多条短信，因此使用数组 msg 比较好。

(6) 注册广播接收器，添加权限。打开根目录下的 AndroidManifest. xml 文件，并编辑之。其代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "cn.com.sgmsc.SMS"
    android:versionCode = "1"
    android:versionName = "1.0" >

    <application
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name" >
        <activity
            android:label = "@string/app_name"
            android:name = ".SMSSendActivity" >
            <intent - filter >
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </activity>
     <receiver android:name = ".SMSReceiver" android:enabled = "true" >
        <intent - filter>
            <action android:name = "android.provider.Telephony.SMS_RECEIVED" />
        </intent - filter>
      </receiver>
   </application>
   <uses - sdk android:minSdkVersion = "10" />
   <uses - permission android:name = "android.permission.SEND_SMS"/>
   <uses - permission android:name = "android.permission.RECEIVE_SMS"/>

</manifest>
```

【运行结果】 在 Eclipse 中启动两个 Android 模拟器，其模拟器号分别为 5554 和 5556。然后运行 SMSDemo 项目。如果系统选用 5556 模拟器运行程序，在其屏幕上可输入对方的手机号，在本例中使用的是 5554；再输入短信内容，如图 11-5 所示，然后单击 SEND 按钮，将短信发送到指定方的手机上。由于短信内容超过了 70 个字符，所以这时将从 5554 模拟器的状态栏中看到接收到两条来自 15555215556 的短信通知，如图 11-6 所示。双击该通知条目可以打开浏览其详细内容，并进行回复，如图 11-7 所示。当回复信息发出后，5556 模拟器随即收到回复信息，显示收到短信的消息提示，如图 11-8 所示。

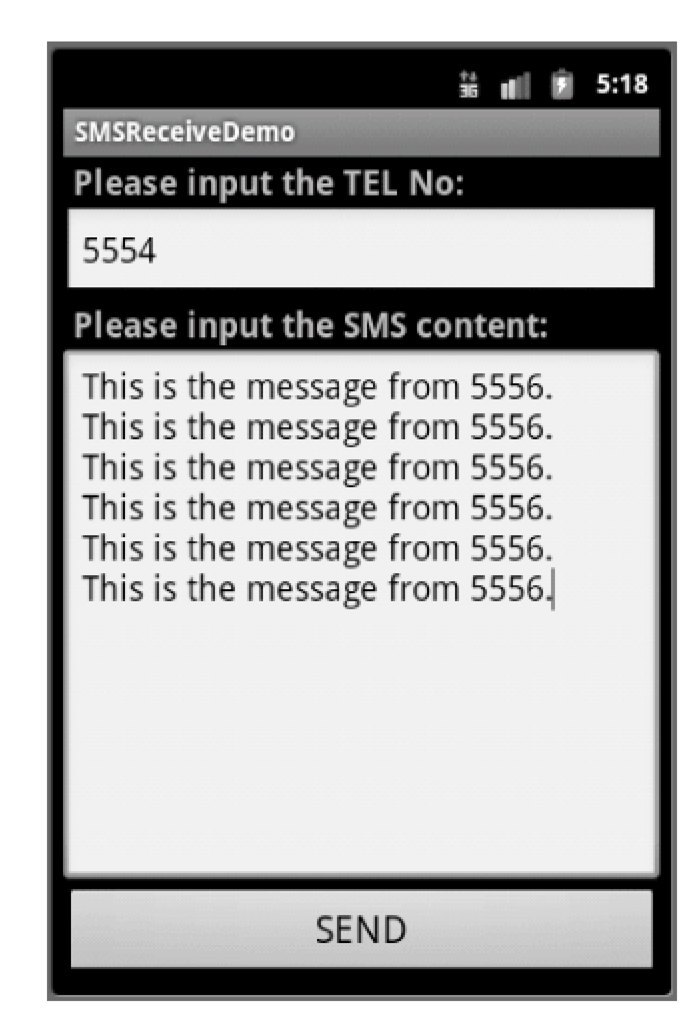


图 11-5　从 5556 模拟机上发出短信

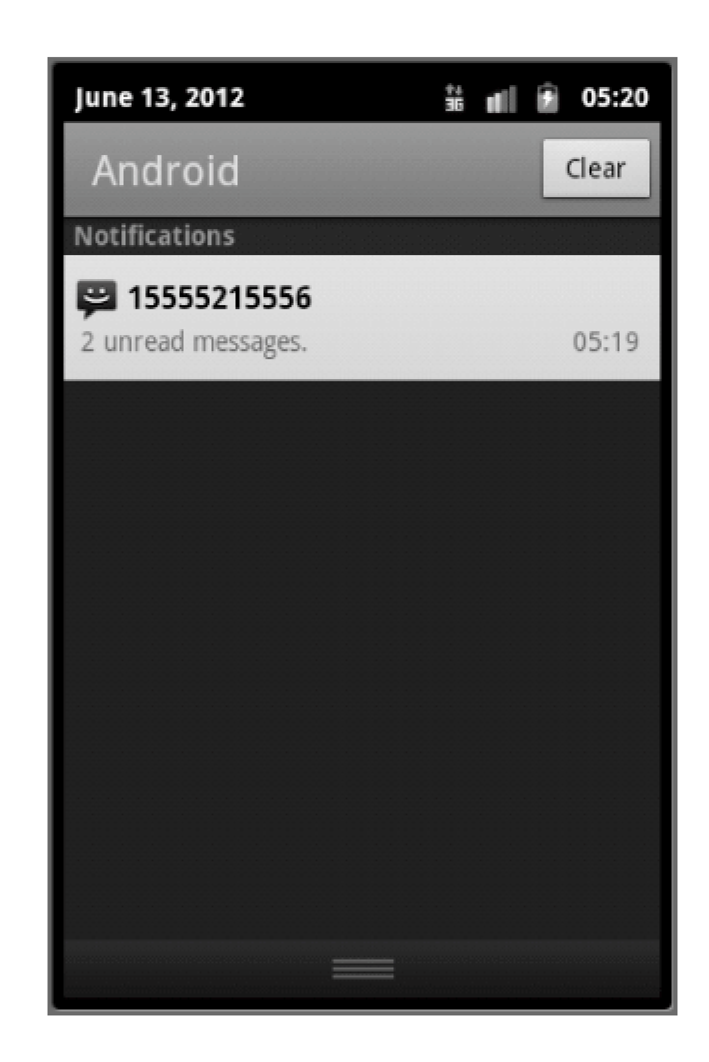


图 11-6 在 5554 模拟机上拉下状态栏

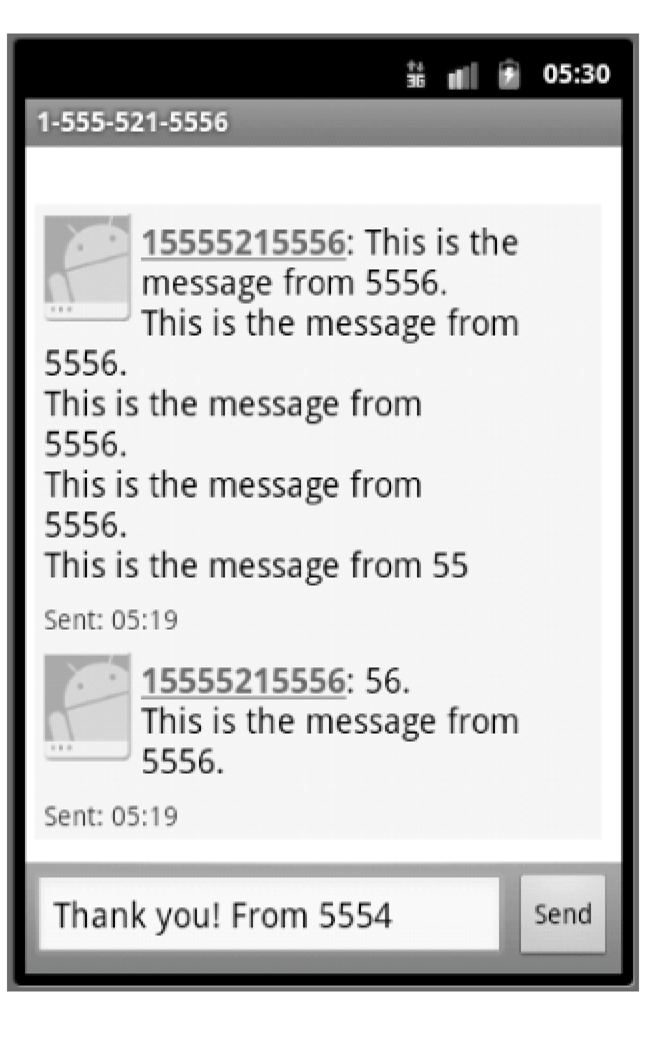


图 11-7 浏览 5554 模拟机上的短信并回复



图 11-8 5556 模拟机收到短信提示

11.2 电话控制

手机的电话功能是最基本的功能。在应用程序中,特别是通信类的应用程序,常做的事情是对电话进行控制、监听和服务,如能够监听电话的呼入,以便执行暂停音乐播放器,当电话结束之后,再恢复播放,或对电话进行录音,或在应用程序中直接拨打电话等。在 Android 平台可以通过 TelephonyManager 和 PhoneStateListener 来完成此任务。

11.2.1 拨打电话

开发拨打电话应用,可以直接调用系统的 dialer 拨号程序。使用 Intent 来启动本地的拨号程序的方式可以是 Intent. ACTION_CALL,也可以是 Intent. ACTION_DIAL。

以使用 Intent. ACTION_CALL 方式为例,如果按钮 bDial 是拨打键,EditText 控件 et 内是要拨打的电话号码,那么在用户按完电话号码后单击该按钮,这个按钮的 OnClickListener 监听的代码如下。

```
Button bDial = (Button)this.findViewById(R.id.Button_dial);
bDial.setOnClickListener(//为拨号按钮添加监听器
        //OnClickListener 为 View 的内部接口,其实现者负责监听鼠标点击事件
        new View.OnClickListener(){
            public void onClick(View v){
                //获取输入的电话号码
                EditText et = (EditText)findViewById(R.id.EditText01);
                String num = et.getText().toString();
                //根据获取的电话号码创建 Intent 拨号
                Intent dial  =  new Intent();
                dial.setAction("android.intent.action.CALL");
                dial.setData(Uri.parse("tel://" + num));
                startActivity(dial);   //激活打电话的 Activity
            }
```

```
        }
);
```

拨打电话代码程序所引入的类都是以前学习过的,并没有引入新类。但是需要在AndroidManifest.xml文件中为应用程序添加打电话的权限:

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

11.2.2 监听电话的状态

Android中所有与电话相关的API都由TelephonyManager类来管理,TelephonyManager类位于android.telephony包下。TelephonyManager作为一个Service接口提供给用户查询电话相关的内容,例如IMEI(国际移动装备辨识码),LineNumber1等。

在TelephonyManager类中有以下三种状态。

(1) TelephonyManager.CALL_STATE_IDLE——待机状态,值为0。

(2) TelephonyManager.CALL_STATE_RINGING——来电状态(振铃),值为1。

(3) TelephonyManager.CALL_STATE_OFFHOOK——通话中(摘机),值为2。

对电话状态进行监听,需要使用PhoneStateListener类。该类也位于android.telephony包下。监听电话相关的事件诸如:响铃、挂断、基站位置变化、语音邮件、呼叫转移等。PhoneStateListener类常用的监听通话方法如表11-2所示。

表11-2 PhoneStateListener类监听通话方法及对应的常量说明

方　法	静态常量	说　明
onCallForwardingIndicatorChanged(boolean cfi)	LISTEN_CALL_FORWARDING_INDICATOR	监听通话转移指示变化回调方法和常量
onCallStateChanged(int state, String incomingNumber)	LISTEN_CALL_STATE	监听呼叫状态变化的回调方法和常量
onCellLocationChanged(CellLocation location)	LISTEN_CELL_LOCATION	监听设备单元位置变化的回调方法和常量
onDataActivity(int direction)	LISTEN_DATA_ACTIVITY	监听数据流量移动方向变化的回调方法和常量
onDataConnectionStateChanged(int state)	LISTEN_DATA_CONNECTION_STATE	监听数据连接状态的变化
onMessageWaitingIndicatorChanged(boolean mwi)	LISTEN_MESSAGE_WAITING_INDICATOR	监听消息等待指示变化的回调方法和常量
onServiceStateChanged(ServiceState serviceState)	LISTEN_SERVICE_STATE	监听网络服务状态变化的回调方法和常量
onSignalStrengthChanged(int asu)	LISTEN_SIGNAL_STRENGTH	监听网络信号强度变化的回调方法和常量

对电话状态进行监听时,还需要在AndroidManifest.xml中为应用程序添加读电话状态的权限:

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

在Android应用中,如果想要监听电话的拨打状况,需要实现以下步骤。

(1) 获取电话服务管理器实例。

```
TelephonyManager telephonyMgr = (TelephonyManager)
                                this .getSystemService(Context.TELEPHONY_SERVICE);
```

(2) 通过 TelephonyManager 设置要对电话事件的监听。例如监听呼叫状态改变。

```
telephonyMgr.listen(new TeleListener (),PhoneStateListener.LISTEN_CALL_STATE);
```

如果需要设置多个监听,可以在 listen()方法的第二个参数上多加几个静态常量,用"|"隔开。例如设置呼叫状态改变和网络服务状态改变监听:

```
telephonyMgr.listen(new TeleListener (),PhoneStateListener.LISTEN_CALL_STATE |
                                  PhoneStateListener. LISTEN_SERVICE_STATE);
```

并且 TeleListener 需要实现父类的几个方法:onCallStateChanged()、onServiceStateChanged()。

如果要取消对电话事件的监听:

```
telephonyMgr.listen(TeleListener, PhoneStateListener.LISTEN_NONE);
```

(3) 通过继承 PhoneStateListener 来重写与第(2)步的第二个参数对应的回调方法,在这些方法中定制自己的规则。

(4) 这一步很重要,就是给应用添加权限:android. permission. READ_PHONE_STATE 。

下面通过一个案例来说明监听电话状态的用法。

【案例 11.3】 开发一个对电话呼叫状态进行监听的应用,并将状态记录下来显示在屏幕上。

【说明】 该案例使用系统自带的拨打电话应用程序进行拨打、接听、挂起等操作。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 TeleCallListener 的 Android 项目。其应用程序名为 TeleCallListener,包名为 cn. com. sgmsc. TelListener,Activity 组件名为 TeleCallListenerActivity。

(2) 设计布局。编写 res/layout 目录下的 main. xml 文件,只在屏幕上放置一个 TextView 控件,其 ID 名为 tv1。在此省略其代码。

(3) 开发逻辑代码。打开 src/cn. com. sgmsc. TelListener 包下的 TeleCallListenerActivity. java 文件,并编辑之。其代码如下所示。

```
package cn.com.sgmsc.TelListener;

import android.app.Activity;
import android.os.Bundle;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import android.widget.TextView;

public class TeleCallListenerActivity extends Activity  {
 TelephonyManager telmanager ;
 String result = "监听电话状态: \n\n";
  TextView textView ;
     @Override
     public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取电话服务
        telmanager = (TelephonyManager) this.getSystemService(TELEPHONY_SERVICE);
        //手动注册对 PhoneStateListener 中的 listen_call_state 状态进行监听
```

```
20          telmanager.listen(new MyCallStateListener(), PhoneStateListener.LISTEN_CALL_STATE);
21
22          textView  = (TextView) findViewById(R.id.tv1);
23          textView.setText(result);
24   }
25     /* 继承 PhoneStateListener 类,可以重写其内部的各种监听方法
26      *然后通过手机状态改变时,系统自动触发这些方法来实现想要的功能
27      */
28     class MyCallStateListener extends PhoneStateListener{
29        @Override
30        public void onCallStateChanged(int state, String incomingNumber) {
31             switch (state) {
32             case TelephonyManager.CALL_STATE_IDLE:
33                 result += "空闲。\n";
34                 break;
35             case TelephonyManager.CALL_STATE_RINGING:
36                 result += "    响铃,来电号码:" + incomingNumber + "。\n";
37                 break;
38             case TelephonyManager.CALL_STATE_OFFHOOK:
39                 result += "挂起。\n";
40             default:
41                 break;
42             }
43             textView.setText(result);
44             super.onCallStateChanged(state, incomingNumber);
45         }
46     }
47
48 }
```

① 第 20 行,对 PhoneStateListener 中的 LISTEN_CALL_STATE 状态进行监听,所以在 MyCallStateListener 类中需要重写 onCallStateChanged()方法。

② 第 28～46 行,定义 MyCallStateListener 类,并重写 onCallStateChanged()方法,在该方法中将 TelephonyManager 的状态用用户可理解的文字显示在 TectView 控件中。

(4) 添加权限。打开根目录下的 AndroidManifest.xml 文件,并在<manifest>标签中加入权限:

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

【运行结果】 在 Eclipse 中启动两个 Android 模拟器,其模拟器号分别为 5554 和 5556。然后运行 TeleCallListener 项目。如果选用 5554 模拟器运行程序,初始运行时屏幕显示效果如图 11-9 所示,这时在 5556 模拟器中启用拨打电话应用拨打号码为“5554”,则 5554 模拟器会做出应答,显示来电,可接通来电,并挂起片刻后再恢复通话,然后挂断。返回到本例的界面,如图 11-10 所示。



图 11-9 5554 模拟器初始运行界面

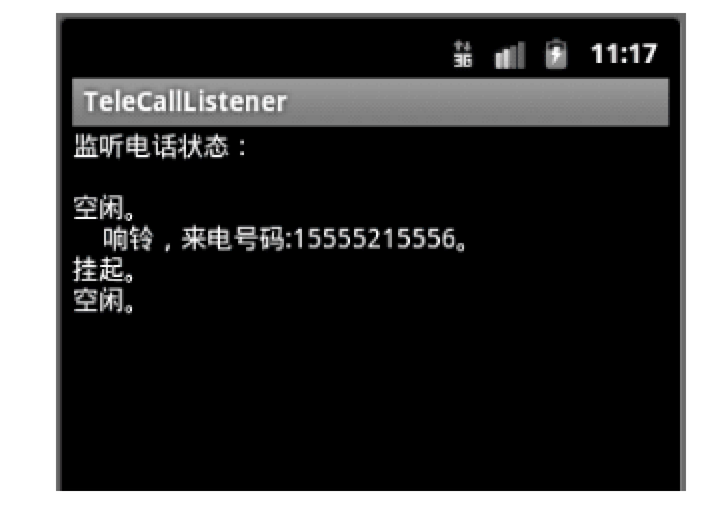


图 11-10 执行完接听、挂起、挂断操作后

11.3 E-mail 功能开发

现在手机功能越来越强大，使用手机发 E-mail 已是一种广泛的应用。Google 在发表 Android 手机平台时，强调的是超强大的网络支持能力，因此，无论通过 GPRS、3G 的电信网络或者是 WIFI 的 WLAN，都能够发 E-mail。

在 Android 平台的底层是采用 SMTP 进行通信的，SMTP（Simple Mail Transfer Protocal）即简单邮件传输协议，由它来控制信件的中转发送方式，并通过 Android 中内置的 Gmail 程序完成邮件发送。发送邮件中使用的 Intent 行为为 android. content. Intent. ACTION_SEND。

E-mail 功能开发非常简单，在这里只给出发送 E-mail 的关键代码段。

```
Intent intent = new Intent(android.content.Intent.ACTION_SEND);//创建 Intent
intent.setType("plain/text");                                 //设置邮件格式
strEmailReciver = new String[]{text01.getText().toString()};  //将收件人放入 Intent 中
strEmailBody = text02.getText().toString();
intent.putExtra(android.content.Intent.EXTRA_EMAIL, strEmailReciver); //设置收件人
intent.putExtra(android.content.Intent.EXTRA_TEXT, strEmailBody);     //设置内容
startActivity(Intent.createChooser(intent, getResources().getString(R.string.str_message)));
                                                                       //打开 Gmail 发邮件
```

11.4 手机特有特性开发

手机特有的特性包括：响应系统设置更改事件，手机的来电提醒设置，音量调节，SIM 卡和电信网络信息，手机电池电量等。在 Android 系统中，有些手机特有的特性设置的变化会影响到应用程序的执行。

11.4.1 系统设置更改事件

在 Android 中，系统设置信息封装在一个 Configuration 类中。Configuration 类位于 android. content. res 包下，主要用来描述与能让应用程序获取的资源相关的所有硬件配置信息。例如系统字体大小，方向，输入设备类型等。

在应用程序中，Android 的默认行为不是总是令人满意的，尤其当配置变化（如屏幕方向和键盘可视）、用户旋转设备或划出键盘等。可以通过监测和响应定制自己的应用程序来对这些变化做出响应。

1. 设置 configChanges 属性

为了能让 Activity 能监听实时的配置变化，需要在 AndroidManifest. xml 文件中的<activity>标签内添加“android:configChanges”属性，其属性值用于指定应用程序要处理的配置变化事件。常用的属性值如下。

（1）orientation——屏幕在纵向和横向间旋转。

（2）keyboardHidden——键盘显示或隐藏。

(3) fontScale——用户变更了首选的字体大小。

(4) locale——用户选择了不同的语言设定。

(5) keyboard——键盘类型变更,例如手机从 12 键盘切换到全键盘。

(6) touchscreen 或 navigation ——触屏或导航方式变化,一般不会发生这样的事件。

在该属性中可以选择多个值,用"|"分隔。例如:

```
android:configChanges = "orientation|keyboard|keyboardHidden"
```

添加上述属性的含义是表示在改变屏幕方向、弹出软件盘和隐藏软键盘时,不再去执行 onCreate()方法,而是直接执行 onConfigurationChanged()方法。如果不声明此段代码,按照 Activity 的生命周期,当改变屏幕方向、弹出软件盘和隐藏软键盘时,都会去执行一次 onCreate()方法,而 onCreate()方法通常会做一些初始化工作。这样,往往会降低程序效率,或需要做许多工作来避免因初始化而造成已得到的数据的丢失,给程序的编码带来逻辑的复杂性。

另外,为了让 Android API 允许用户改变配置信息,还必须在 AndroidManifest.xml 的文件中添加一权限:

```
<uses-permission android:name = "android.permission.CHANGE_CONFIGURATION" />
```

2. 响应 Configuration 的变化

很多时候配置改变(Configuration Change)了,例如经常遇到的屏幕横向纵向进行切换,一般会重新加载 Activity,这样切换起来会看到界面非常的闪。如果使用了 onConfigurationChanged()方法就可以很好地解决这个问题。

如果期望在代码中响应系统设置的变化,可以通过重写 Activity 类的 onConfigurationChanged()方法来实现。在 onConfigurationChanged()方法内,根据 Configuration 的不同配置值进入不同的事件中进行处理。Configuration 类常用的系统设置有如下几个方面。

(1) 屏幕的朝向。可取的值有 ORIENTATION_PORTRAIT(值为 1)、ORIENTATION_LANDSCAPE(值为 2)、ORIENTATION_SQUARE(值为 3)。

(2) 键盘显示或隐藏。可取的值有 KEYBOARDHIDDEN_NO(值为 1)、KEYBOARDHIDDEN_YES(值为 2)。

(3) 键盘类型。可取的值有 KEYBOARD_NOKEYS(值为 1)、KEYBOARD_QWERTY(值为 2)、KEYBOARD_12KEY(值为 3)。

(4) 滚动球方式。可取的值有 NAVIGATION_NONAV(值为 1)、NAVIGATION_DPAD(值为 2)、NAVIGATION_TRACKBALL(值为 3)、NAVIGATION_WHEEL(值为 4)

(5) 触屏模式。可取的值有 TOUCHSCREEN_NOTOUCH(值为 1)、TOUCHSCREEN_STYLUS(值为 2)、TOUCHSCREEN_FINGER(值为 3)。

如果在 AndroidManifest.xml 文件中,对 activity 标签的 configChanges 属性设置了多个值,那么 onConfigurationChanged()方法就可以捕获多个事件。例如,在 AndroidManifest.xml 文件中有如下属性:android:configChanges="orientation|keyboardHidden",则在 onConfigurationChanged()方法中有如下代码片段。

```
@Override
```

```
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    //…… 基于资源值更新 UI 的代码
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        //…… 对不同的屏幕方向做出处理的代码
    }
    if (newConfig.keyboardHidden == Configuration.KEYBOARDHIDDEN_NO) {
        //…… 对键盘可见做出处理的代码
    }
}
```

下面通过一个简单案例来说明响应 Configuration 的变化用法。

【案例 11.4】 响应屏幕朝向的改变,并给出变化的消息提示。

【说明】 在屏幕上加一个背景图片,可以更好地表现屏幕的朝向效果,并且让某些控件在初始运行时不可见,能更好地体现当系统设置改变后,不用执行 onCreate()方法,而是执行 onConfigurationChanged()方法。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 ConfigOrient 的 Android 项目。其应用程序名为 Cfg_Orient,包名为 cn.com.sgmsc.Ori,Activity 组件名为 Cfg_OrientActivity。

(2) 准备图片。将作为背景图片的文件复制到 res/drawable-mdpi 目录中。

(3) 准备字符串资源。编写 res/values 目录下的 strings.xml 文件,代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Cfg_OrientActivity!</string>
    <string name="app_name">Cfg_Orient</string>
    <string name="btn">点击更改屏幕朝向</string>
    <string name="et">当前屏幕朝向:</string>
    <string name="orientP">纵向(PORTRAIT)</string>
    <string name="orientL">横向(LANDSCAPE)</string>
</resources>
```

(4) 设计布局。编写 res/layout 目录下的 main.xml 文件,代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bg"
    >                                   <!-- 声明一个线性布局 -->
    <Button
        android:id="@+id/btn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn"
        />                              <!-- 声明一个 Button 控件 -->
   <EditText
        android:id="@+id/et"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:cursorVisible="false"
```

```
        android:text = "16sp"
        />                          <! -- 声明一个 EditText 控件 -->
</LinearLayout>
```

(5) 开发逻辑代码。打开 src/cn.com.sgmsc.Ori 包下的 Cfg_OrientActivity.java 文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.Ori;

import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Cfg_OrientActivity extends Activity {
 EditText et;
 @Override
 public void onCreate(Bundle savedInstanceState) {      //重写 onCreate()方法
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);                       //设置当前屏幕
   Button btn = (Button)findViewById(R.id.btn);
   et = (EditText)findViewById(R.id.et);
   et.setVisibility(View.GONE);
   btn.setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View v) {
           if(Cfg_OrientActivity.this.getRequestedOrientation() == -1){
                                                        //判断是否可以获得屏幕方向属性
               Toast.makeText(Cfg_OrientActivity.this, "系统的屏幕方向无法获取!!", Toast.
                           LENGTH_LONG).show();
           }
           else{
               if(Cfg_OrientActivity.this.getRequestedOrientation() ==
                  ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE){
                     Cfg_OrientActivity.this.setRequestedOrientation
                                 (ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
               }
               else if(Cfg_OrientActivity.this.getRequestedOrientation() ==
                    ActivityInfo.SCREEN_ORIENTATION_PORTRAIT){
                     Cfg_OrientActivity.this.setRequestedOrientation
                                 (ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
               }
           }
       }
   });
 }
 @Override
 public void onConfigurationChanged(Configuration newConfig) {
      Toast.makeText(this, "系统的屏幕方向发生改变", Toast.LENGTH_LONG).show();
      updateEditText();                                 //更新 EditText 显示的内容
      super.onConfigurationChanged(newConfig);
```

```
48  }
49  public void updateEditText(){
50        int ori = getRequestedOrientation();          //获取屏幕朝向
51        et.setText(getResources().getText(R.string.et));
52        et.setVisibility(View.VISIBLE);
53        switch(ori){                                  //判断屏幕当前朝向
54        case ActivityInfo.SCREEN_ORIENTATION_PORTRAIT:
55              et.append(getResources().getText(R.string.orientP));
56              break;
57        case ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE:
58              et.append(getResources().getText(R.string.orientL));
59              break;
60        }
61   }
62 }
```

① 第 20 行，设置 EditText 不显示。

② 第 21～41 行定义按钮的 OnClickListener 监听。在 onClick()回调事件中，首先判断是否可以获得 requestedOrientation 属性，如果可以，则调用 setRequestedOrientation()方法改变屏幕的朝向。

③ 第 44～48 行，定义 onConfigurationChanged()，在其中调用自定义方法 updateEditText()(在第 49～61 行定义)来显示屏幕朝向改变消息和提示信息。

④ 第 52 行让 EditText 可见，第 55、58 行向 EditText 中追加字符串信息。

(6) 添加权限属性。打开根目录下的 AndroidManifest.xml 文件，添加允许修改系统设置权限和 configChanges 属性，其代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
        package = "cn.com.sgmsc.Ori"
        android:versionCode = "1"
        android:versionName = "1.0">
    <uses - sdk android:minSdkVersion = "10" />
    <uses - permission android:name = "android.permission.CHANGE_CONFIGURATION" />

    <application android:icon = "@drawable/icon" android:label = "@string/app_name">
        <activity android:name = ".Cfg_OrientActivity"
                  android:label = "@string/app_name"
                  android:screenOrientation = "portrait"
                  android:configChanges = "orientation">
            <intent - filter>
              <action android:name = "android.intent.action.MAIN" />
              <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </activity>

    </application>
</manifest>
```

【运行结果】 在 Eclipse 中启动一个 Android 模拟器，然后运行 ConfigOrient 项目。初始运行时屏幕显示效果如图 11-11 所示，这时单击“点击更改屏幕朝向”按钮，屏幕改为横向显示，如图 11-12 所示；再次单击“点击更改屏幕朝向”按钮，屏幕还原为纵向显示，如图 11-13 所示。请注意，图 11-11 与图 11-13 是有区别的。



图 11-11　初始运行时项目的界面



图 11-12　单击按钮之后变为横向显示



图 11-13　再次单击按钮恢复为纵向显示

11.4.2　振动设置

手机一般用振铃和振动来做提醒功能。在应用程序中也可以适当地运用振动特性。在Android 平台下不仅可以启动手机振动，还可以设置振动的周期、持续的时间等参数。

Vibrator 类是控制管理手机振动的类，该类位于 android. os 包下。如果想让手机启动振动，需要创建 Vibrator 对象，创建一个 Vibrator 对象需要调用 getSystemService()方法，如：

```
Vibrator vibrator = (Vibrator)getSystemService(Service.VIBRATOR_SERVICE);
```

在创建语句中，使用了 Service. VIBRATOR_SERVICE 参数，这需要引入 android. app. Service 类。

启动或关闭手机振动，是由 Vibrator 对象中的方法实现的，Vibrator 对象常用的方法如表 11-3 所示。

表 11-3　Vibrator 对象常用方法及说明

方　法	参　数	说　明
vibrate(long[] pattern, int repeat)	pattern：数组，该数组中的第一个元素值是等待多长时间才启动振动，之后的元素值是开启和关闭振动的持续时间，单位以毫秒计。 repeat：重复次数，当其为－1 时，表示不重复，只以 pattern 的方式运行一次	根据指定的模式进行振动
vibrate(long milliseconds)	milliseconds：振动持续的时间，单位以毫秒计	启动振动，并指定持续振动的时间
cancel()		关闭振动

从表中可看到，启动手机振动有两种方式，一种是指定振动模式的启动，另一种是指定振动持续时间的启动。例如，指定振动模式的启动代码为：

```
vibrator.vibrate(new long[]{1000,500,500,1000}, -1);
```

该启动模式表示，第一个参数是一个数组，它定义振动模式为：等待 1s 启动振动，持续振动 0.5s 后停止，停止 0.5s 后重新启动，持续振动 1s。第二个参数为－1，表示只按第一参数指定的振动模式进行一次，不重复。

关闭振动，使用 cancel()方法即可。

最后，请读者记住，要使用振动应用开发，必须在 AndroidManifest.xml 文件中添加声明振动的权限。代码如下：

```
<uses - permission android:name = "android.permission.VIBRATE" />
```

11.4.3 音量调节

一些 Android 应用会自带背景音乐，例如游戏或者一款书本阅读器。那么就需要有调节声音大小或者改变声音模式的控制。手机音量调节由 AudioManage 类实现，该类位于 android.media 包下，该类中包含很多对声音模式和音量控制的方法。常用的对音量进行控制的方式如表 11-4 所示。

表 11-4 AudioManage 类常用的方法及说明

方法	参数	说明
adjustStreamVolume(int stream Type，int direction，int flags)	streamType：声音类型； direction：调整音量的方向； flags：可选的标志位	调整指定声音类型的音量
setMode(int mode)	mode：声音模式	设置声音模式
setRingerMode(int ringerMode)	ringerMode：铃声模式	设置铃声模式
setStreamMute (int streamType，boolean state)	streamType：声音类型； state：是否使该类型声音静音的标志位	设置指定类型的声音是否需要静音

其中，参数 streamType，声音类型可取值有 STREAM_AI_ALARM、STREAM_DTMF、STREAM_MUSIC、STREAM_NOTIFICATION、STREAM_RING、STREAM_SYSTEM 和 STREAM_VOICE_CALL；参数 direction，调整音量的方向可取值有 ADJUST_LOWER、ADJUST_RAISE 和 ADJUST_SAME；参数 flags，可选的标志位可取值有 FLAG_ALLOW_RINGER_MODES、FLAG_PLAY_SOUND、FLAG_REMOVE_SOUND_AND_VIBRATE、FLAG_SHOW_UI 和 FLAG_VIBRATE；参数 mode，声音模式可取值有 NORMAL、RINGTONE 和 IN_CALL；参数 ringerMode，铃声模式可取值有 RINGER_MODE_NORMAL、RINGER_MODE_SILENT 和 RINGER_MODE_VIBRATE。

播放音乐的应用项目，其声音资源文件一般都存放在 res 目录下的 raw 目录下。使用 AudioManager 进行音量控制，有下列几段关键代码。

(1) AudioManager 对象的创建和音量控制。

```
//音量控制,初始化定义
AudioManager mAudioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
//最大音量
int maxVolume = mAudioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
//当前音量
int currentVolume = mAudioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
```

(2) 直接控制音量的大小。

```
if(isSilent){
 mAudioManager.setStreamVolume(AudioManager.STREAM_MUSIC, 0, 0);
}else{
     mAudioManager.setStreamVolume(AudioManager.STREAM_MUSIC, tempVolume, 0);
                                                    //tempVolume 为音量绝对值
}
```

(3) 以步长为单位逐增或逐减地控制音量,并弹出系统默认音量控制条。

```
//降低音量,调出系统音量控制
if(flag == 0){
     mAudioManager.adjustStreamVolume(AudioManager.STREAM_MUSIC,AudioManager.ADJUST_LOWER,
                                       AudioManager.FX_FOCUS_NAVIGATION_UP);
}
//增加音量,调出系统音量控制
else if(flag == 1){
     mAudioManager.adjustStreamVolume(AudioManager.STREAM_MUSIC,AudioManager.ADJUST_RAISE,
                                       AudioManager.FX_FOCUS_NAVIGATION_UP);
}
```

11.4.4 获取手机信息

在 11.2 节中提过的 TelephonyManager 类,不仅可以对电话进行控制,还可以通过 TelephonyManager 对象获取手机卡及电信网络等信息。

1. TelephonyManager 类简介

TelephonyManager 类位于 android.telephony 包下,该类提供了一系列用于访问与手机通信相关的状态和信息的 get...()方法。

关于手机通信方面的信息如下。

(1) IMEI(International Mobile Equipment Identity,国际移动装备辨识码),由 15 位数字组成,与每台手机一一对应,而且该码是全世界唯一的。

(2) IMSI(International Mobile Subscriber Identity,国际移动用户识别码),由 15 位码组成,其结构如下:MCC+MNC+MSIN。

(3) MCC 是移动国家码(三位)。

(4) MNC 是移动网络码(两位)。

(5) MSIN 是身份手机用户号(10 位)。

如果要获取手机上的相关信息,首先要创建 TelephonyManager 对象实例,其创建方法与 11.2 节中介绍的一样:Context.getSystemService(Context.TELEPHONY_SERVICE)。并且需要在 AndroidManifest.xml 中添加读取手机信息的权限:

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

下面通过一个案例来说明如何获取手机的通信状态和信息。

2. 获取手机信息

【案例 11.5】 获取手机当前的 SIM 卡、电信网络、运营商、IMEI 和 IMSI 等信息。

【说明】 在案例中介绍 TelephonyManager 常用的 get...()方法。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 MobileInfos 的 Android 项目。其应用程序名为 Mobil_Infos，包名为 cn.com.sgmsc.MobileInfo，Activity 组件名为 Mobil_InfosActivity。

(2) 设计布局。编写 res/layout 目录下的 main.xml 文件，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="20px"
            >

            <TextView android:id="@+id/title"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textSize="20sp"
                android:paddingBottom="8dip"
                android:text="" />

            <TextView android:id="@+id/info"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="" />

        </LinearLayout>
</ScrollView>
```

(3) 开发逻辑代码。打开 src/cn.com.sgmsc.MobileInfo 包下的 Mobil_InfosActivity.java 文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.MobileInfo;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.widget.TextView;

public class Mobil_InfosActivity extends Activity  {

 TextView info,title;

    @Override
    public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.main);
    title = (TextView) findViewById(R.id.title);
    title.setText("运营商信息: ");
         info = (TextView) findViewById(R.id.info);
         info.setText(load_data(this));
```

```
21      }
22
23  private static String load_data(Context cis) {
24          //声明 TelephonyManager 对象的引用
25          TelephonyManager tm = (TelephonyManager) cis.getSystemService(Context.TELEPHONY_SERVICE);
26          String str = "";
27          str += "设备编号(IMEI) = " + tm.getDeviceId() + "\n"; //IMEI: 国际移动设备辨识码
28          str += "设备软件版本: " + tm.getDeviceSoftwareVersion() + "\n";
29          str += "本机号码: " + tm.getLine1Number() + "\n";
30          str += "网络国别: " + tm.getNetworkCountryIso() + "\n";
31          str += "网络运营商代号: " + tm.getNetworkOperator() + "\n";
32          str += "网络运营商名称: " + tm.getNetworkOperatorName() + "\n";
33          str += "网络制式: " + tm.getNetworkType() + "\n";
34          str += "手机制式: " + tm.getPhoneType() + "\n";
35          str += "SIM 卡国别: " + tm.getSimCountryIso() + "\n";
36          str += "SIM 卡运营商代号: " + tm.getSimOperator() + "\n";
37          str += "SIM 卡运营商名称: " + tm.getSimOperatorName() + "\n";
38          str += "SIM 卡序列号: " + tm.getSimSerialNumber() + "\n";
39          str += "SIM 卡状态: " + tm.getSimState() + "\n";
40          str += "语音信箱号码: " + tm.getVoiceMailNumber() + "\n\n";
41          //获取国际移动用户识别码 IMSI 的相关信息:
42          str += "国际移动用户识别码 IMSI 相关信息: \n";
43          str += "SubscriberId(IMSI): " + tm.getSubscriberId() + "\n";
44          int mcc = cis.getResources().getConfiguration().mcc;
45          int mnc = cis.getResources().getConfiguration().mnc;
46          str += "IMSI MCC (Mobile Country Code): " + String.valueOf(mcc) + "\n";
47          str += "IMSI MNC (Mobile Network Code): " + String.valueOf(mnc) + "\n";
48          return str;
49  }
50
51 }
```

第 26～47 行,为字符串变量 str 不断追加字符信息,其信息来自于 get()方法。

(4) 添加权限。打开根目录下的 AndroidManifest.xml 文件,并在<manifest>标签中加入权限:

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

【运行结果】 在 Eclipse 中启动一个 Android 模拟器,运行 MobileInfos 项目,屏幕显示效果如图 11-14 所示。



图 11-14 MobileInfos 项目的运行结果

11.4.5 获取手机电池电量

获取手机的电池电量信息是应用程序中经常需要的应用,在 Android 系统中,从系统中获取手机电池电量发生变化的信息通常是通过 Intent 广播来实现的,这个 Intent 常用的 Action 有 Intent.ACTION_BATTERY_CHANGED(电池电量发生改变时)、Intent.ACTION_BATTERY_LOW(电池电量达到下限时)和 Intent.ACTION_BATTERY_OKAY(电池电量从低恢复到高时)。并且需要为应用程序注册

BroadcastReceiver 组件，以便在电池电量发生变化后，通过该 BroadcastReceiver 组件捕获 ACTION_BATTERY_CHANGED 动作，接收到系统发出相应的广播，来得到电池电量信息，并进行应用程序需要的相应处理。如果要停止监测手机电池电量时，只需注销 BroadcastReceiver 组件即可。

下面通过一个简单案例来说明电量提示应用。

【案例 11.6】 通过 BroadcastReceiver 获取手机电池电量信息。

【说明】 在案例中使用一种开关按钮控件 ToggleButton，用于监测或停止监测手机电池的电量，该控件位于 android. widget 包下。ToggleButton 只能有选中和未选中两种状态，并且需要为不同的状态设置不同的显示文本，属性 android: textOff 为未选中显示文本，属性 android: textOn 为选中显示文本。

本例中，监测或停止监测手机电池电量是由注册或注销 BroadcastReceiver 组件控制，所以在程序中需要动态控制 BroadcastReceiver 组件的注册和注销，而不能在 AndroidManifest. xml 文件中注册。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 BatteryInfo 的 Android 项目。其应用程序名为 BatteryInfo，包名为 cn. com. sgmsc. Batt，Activity 组件名为 BatteryInfoActivity。

(2) 设计布局。编写 res/layout 目录下的 main. xml 文件，代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
  android:orientation = "vertical"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent">

  <ToggleButton android:id = "@ + id/tbtn"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:textOn = "停止获取电量信息"
        android:textOff = "获取电量信息" />

   <TextView android:id = "@ + id/tv"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content" />

 </LinearLayout>
```

(3) 开发逻辑代码。打开 src/cn. com. sgmsc. Batt 包下的 BatteryInfoActivity. java 文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.Batt;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.widget.CompoundButton;
import android.widget.TextView;
```

```
11 import android.widget.ToggleButton;
12 import android.widget.CompoundButton.OnCheckedChangeListener;
13
14 public class BatteryInfoActivity extends Activity {
15      private ToggleButton tb = null;
16      private TextView tv = null;
17      private BatteryReceiver battreceiver  = null;
18
19      @Override
20      public void onCreate(Bundle savedInstanceState) {
21          super.onCreate(savedInstanceState);
22          setContentView(R.layout.main);
23
24          battreceiver = new BatteryReceiver();          //创建 BroadcastReceiver 组件对象
25          tv = (TextView)findViewById(R.id.tv);
26          tb = (ToggleButton)findViewById(R.id.tbtn);
27          tb.setOnCheckedChangeListener(new OnCheckedChangeListener(){
28
29              public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {
30                  if(isChecked){                          //获取电池电量
31                      IntentFilter battfilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
32                      registerReceiver(battreceiver, battfilter);
                                                           //注册 BroadcastReceiver
33                  }else {                                 //停止获取电池电量
34                      unregisterReceiver(battreceiver);//注销 BroadcastReceiver
35                      tv.setText(null);
36                  }
37              }
38          });
39      }
40
41      private class BatteryReceiver extends BroadcastReceiver{
42
43          @Override
44          public void onReceive(Context context, Intent intent) {
45              int current = intent.getExtras().getInt("level");   //获得当前电量
46              int total = intent.getExtras().getInt("scale");     //获得总电量
47              int percent = current * 100/total;
48              tv.setText("现在的剩余电量是: " + percent + " % 。");
49          }
50      }
51
52 }
```

① 第 24 行,创建 BatteryReceiver 对象 battreceiver,这个 BatteryReceiver 类继承自 BroadcastReceiver;第 31 行创建一个 IntentFilter 对象 battfilter,其 action 为 Intent.ACTION_BATTERY_CHANGED;第 32 行注册 BroadcastReceiver。当 battreceiver 接收到系统传过来的电池电量信息时,就执行这个 BroadcastReceiver 的 onReceive()方法。

② 第 41～50 行定义了 BatteryReceiver 类,其中重写了 onReceive()方法。在该方法中取得当前的剩余电量,并把它显示在屏幕上。

【运行结果】 在 Eclipse 中启动一个 Android 模拟器,运行 BatteryInfo 项目,初始运行界面如图 11-15 所示,单击“获取电量信息”按钮后,在按钮下方显示剩余电量,如图 11-16 所示。




图 11-15 运行 BatteryInfo 项目的初始界面



图 11-16 单击“获取电量信息”按钮之后

11.5 手机传感器开发

所谓传感器就是能够探测如光、热、温度、重力、方向等的功能。Android 系统的一大亮点之一就是对传感器的应用，如水平仪、指北针、计步器等。

Android 常用的传感器包括加速度传感器(Accelerometer)、方向传感器(Orientation)、磁场传感器(Magnetic Field)、温度传感器(Temperature)、环境光传感器(Light)、距离传感器(Proximity)、压力传感器(Pressure)、条码传感器(Barcode)，Android SDK 2.3 平台新增了更多类型传感器的支持，包括：陀螺仪(gyroscope)、重力感应器(gravity)、旋转矢量(rotation vector)和线性加速器(linear acceleration)等。Android 用 SensorManager 来管理这些传感器。

11.5.1 传感器管理器

1. SensorManager 类

SensorManager 类位于 android. hardware 包下，该类是用来管理传感器的。在 SensorManager 中，传感器的类型由常量来表示。常用的常量如表 11-5 所示。

表 11-5 SensorManager 中常用的传感器类型常量

类 型 常 量	内 部 值	描 述
Sensor. TYPE_ACCELEROMETER	1	加速度传感器
Sensor. TYPE_MAGNETIC_FIELD	2	磁场传感器
Sensor. TYPE_ORIENTATION	3	方向传感器
Sensor. TYPE_GYROSCOPE	4	陀螺仪传感器
Sensor. TYPE_LIGHT	5	环境光照传感器
Sensor. TYPE_PRESSURE	6	压力传感器
Sensor. TYPE_TEMPERATURE	7	温度传感器
Sensor. TYPE_PROXIMITY	8	距离传感器

SensorManager 类的主要方法有下列几个。

(1) 获取某种传感器的默认传感器方法：

```
Sensor defaultGyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

(2) 获取某种传感器的列表方法：

```
List<Sensor> pressureSensors = sensorManager.getSensorList(Sensor.TYPE_PRESSURE);
```

(3) 获取所有传感器的列表方法：

```
List<Sensor> allSensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
```

对于某一个传感器，它通过 get 方法获取具体信息。常用的 get()方法如表 11-6 所示。

表 11-6 传感器使用的 get()方法及说明

方　法	描　述	方　法	描　述
getMaximumRange()	最大取值范围	getType()	传感器类型
getName()	设备名称	getVentor()	设备供应商
getPower()	功率	getVersion()	设备版本号
getResolution()	精度		

2. SensorEventListener 接口

SensorEventListener 是一个接口，是在传感器值实时更改时，希望接收更新的类要实现的接口。SensorEventListener 接口位于 android. hardware 包下，它是开发传感器应用最主要的工作。应用程序实现该接口来监视硬件中一个或多个可用传感器。实现 SensorEventListener 接口主要需要实现以下两个方法。

1) onAccuracyChanged(int sensor, int accuracy)

该方法在传感器的精确度发生变化时调用，参数包括两个整数：一个表示传感器，另一个表示该传感器新的准确度值。SensorManager 提供了三种精确度，由高到低分别为：SENSOR_STATUS_ACCURACY_HIGH、SENSOR_STATUS_ACCURACY_MEDIUM 和 SENSOR_STATUS_ACCURACY_LOW。

2) onSensorChanged(SensorEvent event)

该方法在传感器的数据发生变化时调用，该方法只有一个 SensorEvent 类型的参数 event，其中 SensorEvent 类有一个 values 变量非常重要，该变量的类型是 float[]。但该变量最多只有三个元素，而且根据传感器的不同，values 变量中元素所代表的含义也不同。有些传感器只提供一个数据值，另一些则提供三个浮点值。如方向和加速度传感器都提供三个数据值。开发传感器应用的主要业务代码应该放在这里执行，如读取数据并根据数据的变化进行相应的操作等。

在 Android 1.5 API Level 3 以前，感应器应用编程使用的是 SensorListener 接口，现在被 SensorEventListener 所代替。它们都是注册一个 Listener，其主要区别在于 onSensorChanged()方法的参数上，SensorListener 接口内使用方法 onSensorChanged (int sensor, float[] values)。

11.5.2 Android 常用传感器

1. 加速度传感器

加速度传感器(Accelerometer)主要感应手机的运动，在注册了传感器监听器后加速度传感器主要捕获三个参数 values[0]、values[1]、values[2]，以 m/s^2 为单位。

values[0]：空间坐标系中 X 轴方向上的加速度减去重力加速度在 X 轴上的分量。
values[1]：空间坐标系中 X 轴方向上的加速度减去重力加速度在 Y 轴上的分量。
values[2]：空间坐标系中 X 轴方向上的加速度减去重力加速度在 Z 轴上的分量。

2. 方向传感器

方向传感器(Orientation)主要感应手机方位的变化，其每次读取的都是静态的状态值，在注册了传感器监听器后方向传感器主要捕获三个参数 values[0]、values[1]、values[2]，分别代表手机沿 Yaw 轴、Pitch 轴和 Roll 轴方向转过的角度。

Yaw 轴，是三个方向轴中唯一不变的轴，其方向总是竖直向上，和空间坐标系中的 Z 轴是等同的，也就是重力加速度 g 的反方向。

Pitch 轴，其方向依赖于手机沿 Yaw 轴的转动情况，即当手机沿 Yaw 轴转过一定角度后，Pitch 轴也相应围绕 Yaw 轴转过相同的角度。Pitch 轴的位置依赖于手机沿 Yaw 轴转过的角度，好比 Yaw 轴和 Pitch 轴是两个焊死在一起成 90°夹角的铁棍。

Roll 轴，其方向虽然依赖于手机沿 Yaw 轴和 Pitch 轴转动的情况，但是 Roll 轴的确定并不复杂，Roll 轴与手机之间的相对关系是固定的。其总是沿着手机屏幕长边正方向，是与手机绑定的。

三个方向轴的关系如图 11-17 所示。

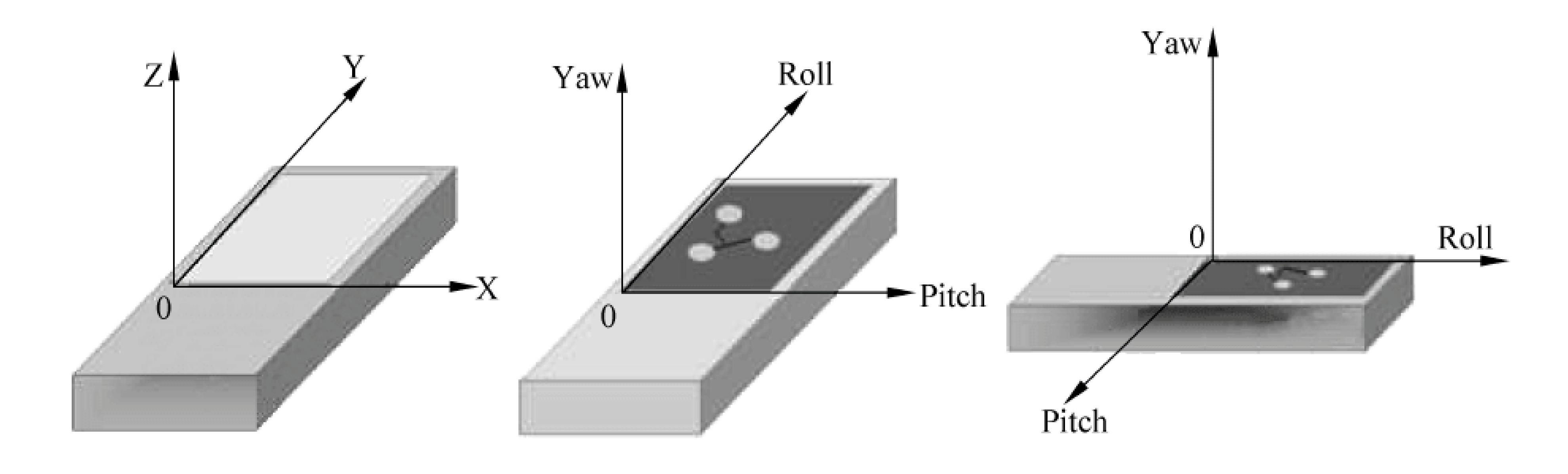


图 11-17　Yaw、Pitch、Roll 方向轴在手机上的位置

3. 磁场传感器

磁场传感器(Magnetic Field)主要用于感应周围的磁感应强度，注册监听器后其主要捕获三个参数：values[0]、values[1]、values[2]。三个参数分别代表磁感应强度在空间坐标系中三个方向轴上的分量，单位为微特拉斯(μT)。通过该传感器可以开发出指北针、罗盘等磁场应用。

4. 温度传感器

温度传感器(Temperature)用于感应周围的温度，注册监听器后只捕获一个参数：values[0]。该参数代表当前的温度。通过运用温度传感器可以开发出手机温度计等应用。

5. 环境光传感器

环境光传感器(Light)用于感应周围的光强，注册监听器后只捕获一个参数：values[0]。

该参数代表周围的光照强度，单位为勒克斯(lux)。

6. 距离传感器

距离传感器(Proximity)用于感应周围靠近的物体，注册监听器后只捕获一个参数：values[0]。该参数代表传感器的物体距离，以厘米(cm)为单位。

11.5.3　传感器应用的开发

1. 开发前的准备工作

在 Android 模拟器中运行传感器应用程序需要借助于 SensorSimulator 工具软件，这个工具是一个开源免费的程序，它可以模拟加速度、方向、磁场、温度、环境光、压力等传感器的场景。其下载网址为 http://code.google.com/p/openintents/wiki/SensorSimulator。为了能支持在 SDK 2.3 版本上运行，本书下载的文件是 sensorsimulator-2.0-rc1.zip 压缩包。下载完成后还需要做一些工作，以便用户能使用 SensorSimulator 工具软件。接下来的操作步骤如下。

(1) 将下载的 sensorsimulator-2.0-rc1.zip 文件解压到磁盘上，在该解压文件夹的 bin 文件夹下可以看到相应的.jar 包和.apk 包文件。本书是将其解压在 E:\sensorsimulator-2.0-rc1 文件夹下。

(2) 在 Eclipse 中启动模拟器。

(3) 向模拟器安装 SensorSimulator。首先使用"cmd"进入命令行状态，输入"E:"并回车，将当前盘符设为 E 盘；然后输入"cd sensorsimulator-2.0-rc1\bin"命令，将 E:\ sensorsimulator-2.0-rc1\bin 文件夹设置为当前文件夹；在此文件夹提示符下输入命令"adb install SensorSimulator-2.0-rc1.apk"，如果运行完这些命令后，出现"Success"信息，说明已经为开启的模拟器安装好 SensorSimulator，如图 11-18 所示。

这时可以在模拟器的应用程序图标中看到 SensorSimulator 应用程序，如图 11-19 所示。

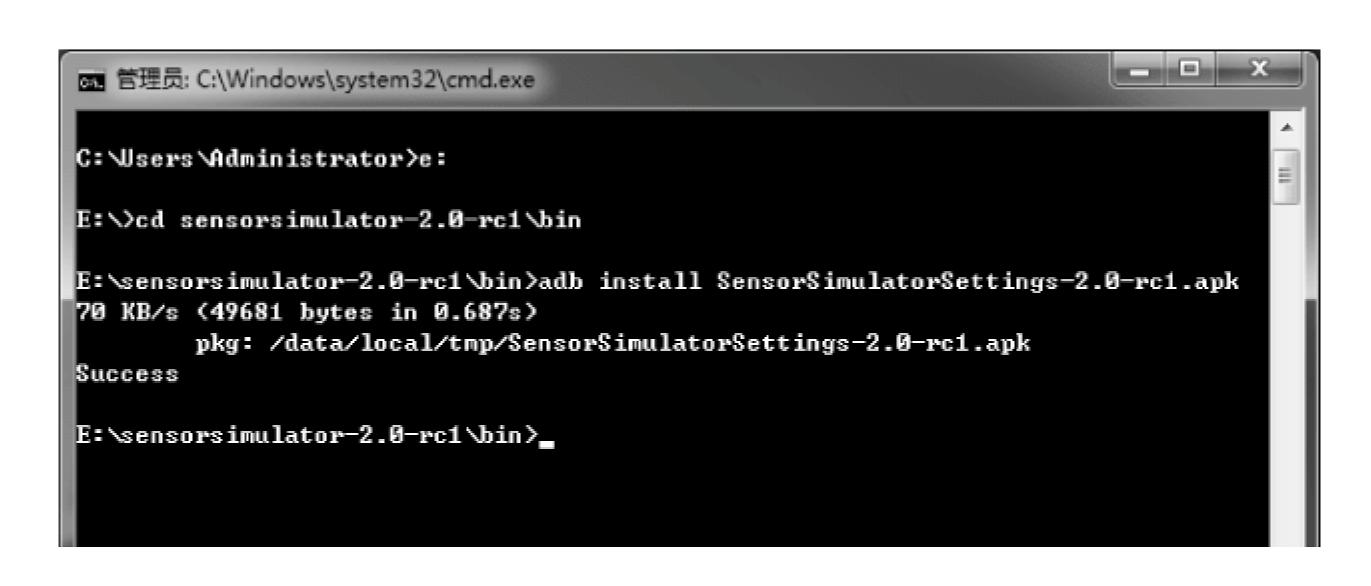


图 11-18　在命令行状态向模拟器安装 SensorSimulator 的过程



图 11-19　在模拟器的应用程序列表中

2. 开发流程

在模拟器上开发传感器应用需要借助于 SensorSimulator 工具，所以其开发流程要比一般的应用多一些操作步骤。

开发一个传感器应用项目，主要流程如下。

1) 新建一个应用项目

在 Eclipse 中新建一个传感器应用项目。创建过程与其他的应用项目一样。

2) 为该项目添加 JAR 包

为该项目添加 JAR 包是开发传感器应用项目必须做的，其他的应用项目不用做。具体操作步骤如下。

(1) 在 Eclipse 的 Project Explorer 中选择该项目，然后右击，在弹出菜单中选择 Properties 菜单项，如图 11-20 所示。

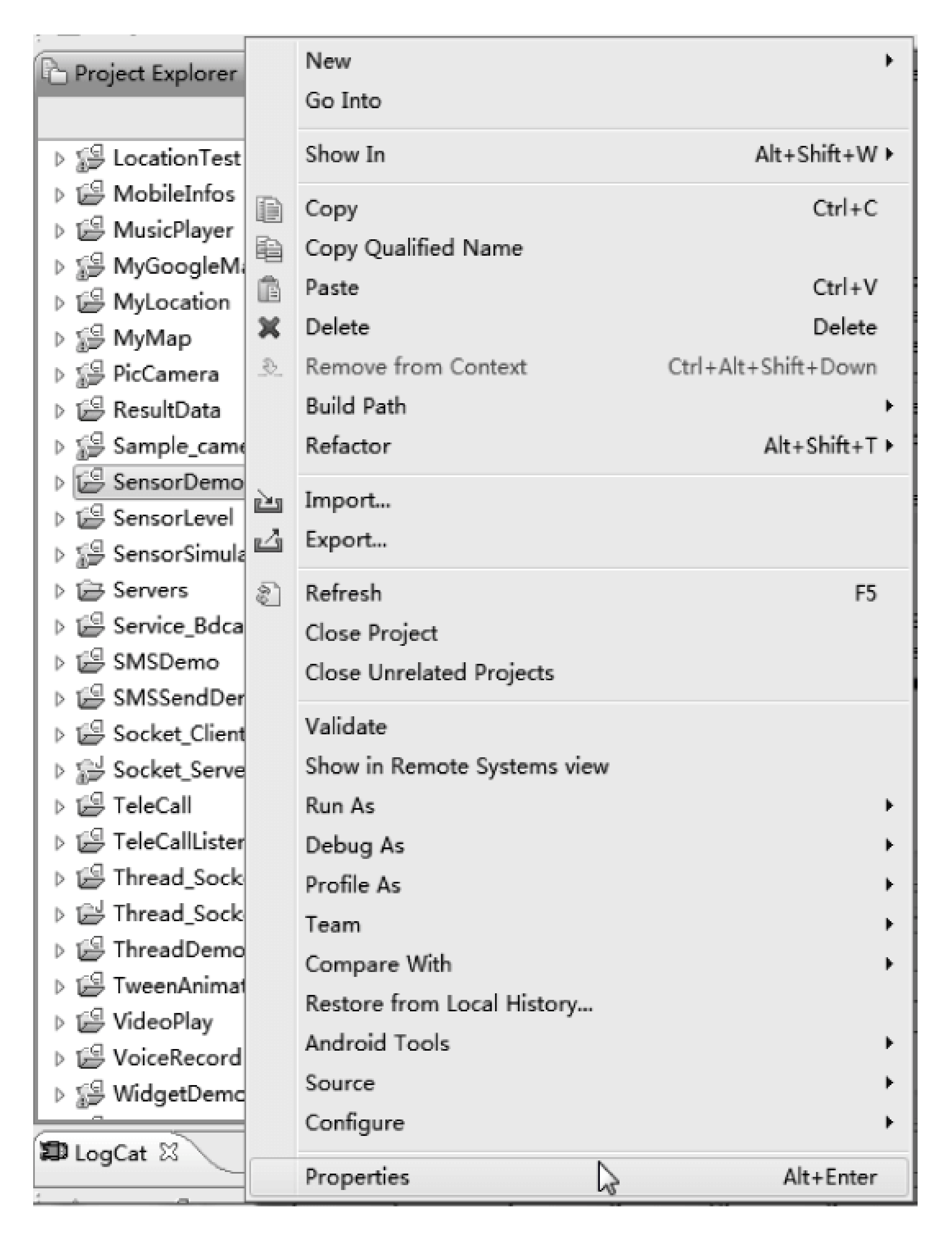


图 11-20　选择该项目的 Properties 菜单项

(2) 进入 Properties for SensorDemo 窗口，先在左侧选择 Java Build Path 选项，再在右侧视窗中选择 Libraries 选项卡，然后单击 Add External JARs...按钮，打开"文件选择"对话框，在 SensorSimulator 的解压文件夹的 lib 子文件夹下选择 sensorsimulator-lib-2.0-rc1.jar 文件，单击"打开"即可，如图 11-21 所示。

(3) 单击 OK 按钮，即完成该项目的 JAR 包添加。

3) 项目资源素材准备

如果该项目需要使用到一些图片、图标、声音音效、文本字符串或颜色等资源，需要预先准备好，存放到项目的指定目录中。

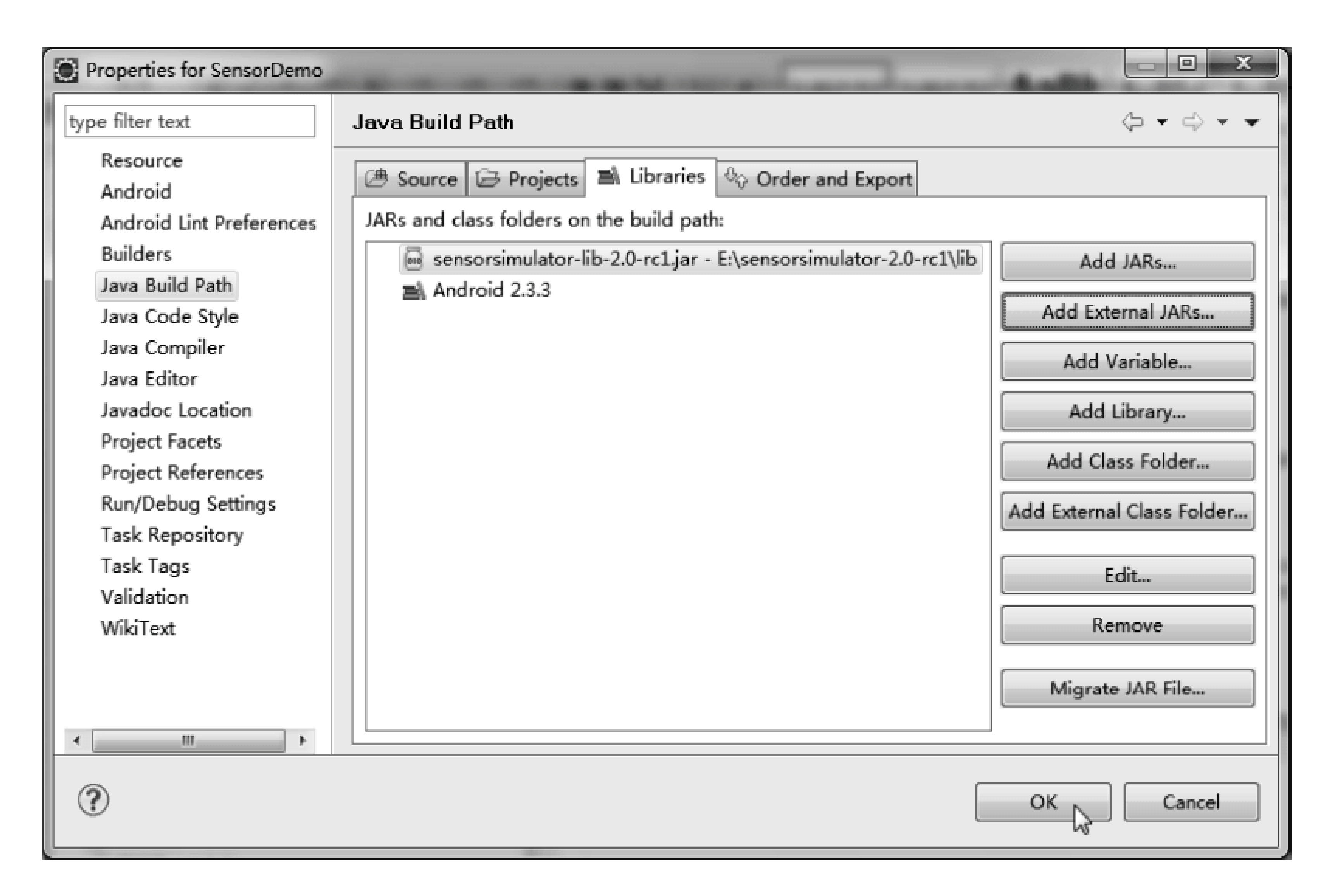


图 11-21 为项目添加了一个 JAR 包

4）项目编码开发

Android 平台下传感器应用的开发通过监听器机制来实现，如果要针对某一种或多种传感器开发应用，主要的步骤如下。

（1）创建 SensorManager 对象。

在模拟器中运行，使用下列代码创建 SensorManager 对象：

```
SensorManagerSimulator mySensorManager = SensorManagerSimulator.getSystemService(this,SENSOR_
SERVICE);
mySensorManager.connectSimulator();
```

在真机中运行，使用下列代码创建 SensorManager 对象：

```
SensorManager mySensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
```

（2）实现 SensorEventListener 接口。

在实现 SensorEventListener 接口中需要实现 onAccuracyChanged(int sensor, int accuracy)和 onSensorChanged(SensorEvent event)两个方法。其主要的代码段如下：

```
private final SensorEventListener mSensorEventLisener = new SensorEventListener (){
    @Override
    public void onAccuracyChanged(int sensor, int accuracy) {
          //加入传感器的精确度发生变化时的处理代码
    }
    public void onSensorChanged(SensorEvent event) {
        //加入传感器的数据发生变化时的处理代码
    }
}
```

（3）注册与注销 SensorListener。

开发完 SensorListener 之后，剩下的工作就是在程序的适当位置注册监听和取消监听了。在这里调用步骤（1）中获得的 SensorManager 对象的 registerListener()方法来注册监听器，其接收的参数为监听器对象、传感器类型以及传感器事件传递的频度。

注销 SensorListener 时调用 SensorManager 的 unregisterListener()方法，一般来讲在编程中，注册和注销的方法应该成对出现，如果在 Activity 的 onResume()方法中注册 SensorListener 监听，就应该在 onPause()方法中注销。

(4) 添加权限。

在 AndroidManifest.xml 文件中添加权限：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

5) 运行传感器项目

在模拟器上运行项目前，必须启动 SensorSimulator 工具。其操作方法是：在 SensorSimulator 的解压文件夹的 bin 文件夹下，运行 sensorsimulator-2.0-rc1.jar 文件，如图 11-22 所示。

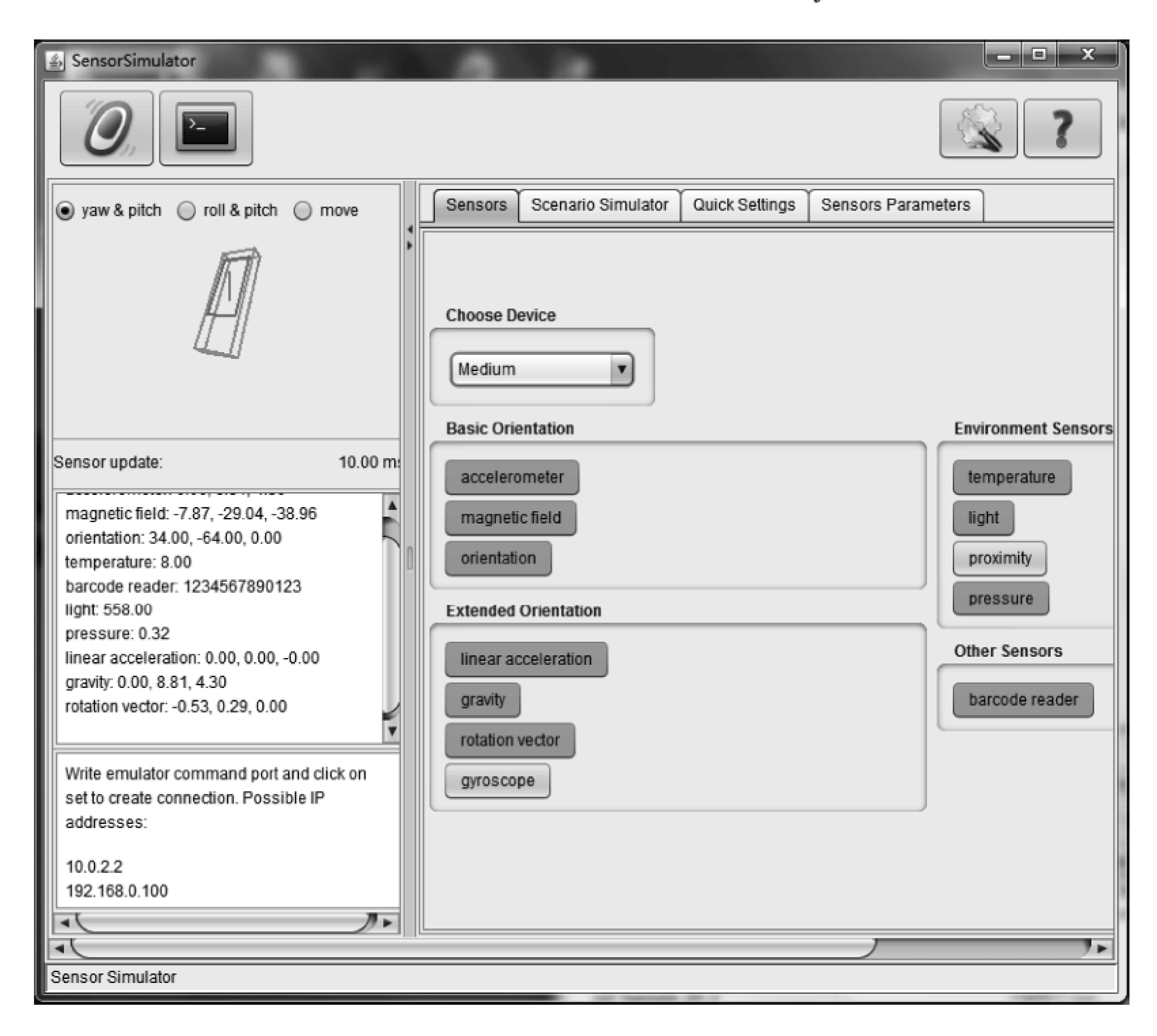


图 11-22 SensorSimulator 工具软件界面

然后在模拟器上运行应用项目。

在 SensorSimulator 窗口的左上角区域，通过鼠标拖动操作，可以模拟手机的方向、位置的旋转或移动，从而可以看到模拟器中运行项目的同步变化效果。

3. 传感器应用案例

【案例 11.7】 实时获取手机支持的各类传感器的数据信息。

【说明】 在案例中获取的一些常用传感器的数据，这些数据也可以在 SensorSimulator 窗口中看到。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 SensorDemo 的 Android 项目。其应用程序名为 SensorDemo，包名为 cn.com.sgmsc.Sensor，Activity 组件名为 SensorDemoActivity。

(2) 添加 JAR 包。为该项目添加 JAR 包。

(3) 准备字符串资源。编写 res/values 目录下的 strings.xml 文件,代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SensorDemoActivity!</string>
    <string name="app_name">SensorDemo</string>
    <string name="Accelerometer">加速度传感器:</string>
    <string name="LinearAcceleration">线性加速器:</string>
    <string name="Gravity">重力传感器:</string>
    <string name="Magnetic_Field">电磁场传感器:</string>
    <string name="Orientation">方向传感器:</string>
   <string name="Temperature">温度传感器:</string>
   <string name="Light">环境光线传感器:</string>
   <string name="Pressure">压力传感器:</string>
   <string name="Rotation_Vector">旋转向量:</string>
   <string name="Barcode">条码传感器:</string>
</resources>
```

(4) 设计布局。编写 res/layout 目录下的 main.xml 文件。在这里使用 RelativeLayout 布局,代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView
        android:id="@+id/text_accelerometer"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/Accelerometer" />
  <TextView
        android:id="@+id/text_linear_acceleration"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/LinearAcceleration" />
  <TextView
        android:id="@+id/text_gravity"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/Gravity" />
  <TextView
        android:id="@+id/text_light"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/Magnetic_Field" />
  <TextView
        android:id="@+id/text_temperature"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/Temperature" />
  <TextView
        android:id="@+id/text_orientation"
```

```
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/Orientation" />
  <TextView
        android:id = "@ + id/text_magnetic_field"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/Magnetic_Field" />
  <TextView
        android:id = "@ + id/text_pressure"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/Pressure" />
  <TextView
        android:id = "@ + id/text_rotation_vector"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/Rotation_Vector" />
  <TextView
        android:id = "@ + id/text_barcode"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/Barcode" />
</LinearLayout>
```

(5) 开发逻辑代码。打开 src/cn. com. sgmsc. Sensor 包下的 SensorDemoActivity. java 文件,并编辑之。其代码如下所示。

```
package cn.com.sgmsc.Sensor;

import org.openintents.sensorsimulator.hardware.Sensor;
import org.openintents.sensorsimulator.hardware.SensorEvent;
import org.openintents.sensorsimulator.hardware.SensorEventListener;
import org.openintents.sensorsimulator.hardware.SensorManagerSimulator;

import android.app.Activity;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class SensorDemoActivity extends Activity {

  private SensorManagerSimulator mSensorManager;

  private TextView mTextViewAccelerometer;
  private TextView mTextViewGravity;
  private TextView mTextViewLinearAcceleration;
  private TextView mTextViewLight;
  private TextView mTextViewTemperature;
  private TextView mTextViewOrientation;
  private TextView mTextViewMagneticField;
  private TextView mTextViewPressure;
  private TextView mTextViewRotationVector;
  private TextView mTextViewBarcode;

```

```
private SensorEventListener mEventListenerAccelerometer;
private SensorEventListener mEventListenerGravity;
private SensorEventListener mEventListenerLinearAcceleration;
private SensorEventListener mEventListenerLight;
private SensorEventListener mEventListenerTemperature;
private SensorEventListener mEventListenerOrientation;
private SensorEventListener mEventListenerMagneticField;
private SensorEventListener mEventListenerPressure;
private SensorEventListener mEventListenerRotationVector;
private SensorEventListener mEventListenerBarcode;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mTextViewAccelerometer = (TextView) findViewById(R.id.text_accelerometer);
    mTextViewGravity = (TextView) findViewById(R.id.text_gravity);
    mTextViewLinearAcceleration = (TextView) findViewById(R.id.text_linear_acceleration);
    mTextViewLight = (TextView) findViewById(R.id.text_light);
    mTextViewTemperature = (TextView) findViewById(R.id.text_temperature);
    mTextViewOrientation = (TextView) findViewById(R.id.text_orientation);
    mTextViewMagneticField = (TextView) findViewById(R.id.text_magnetic_field);
    mTextViewPressure = (TextView) findViewById(R.id.text_pressure);
    mTextViewRotationVector = (TextView) findViewById(R.id.text_rotation_vector);
    mTextViewBarcode = (TextView) findViewById(R.id.text_barcode);

    mSensorManager = SensorManagerSimulator.getSystemService(this,SENSOR_SERVICE);
    mSensorManager.connectSimulator();

    initListeners();                                    //初始化监听器
}

private void initListeners() {
    //创建加速度传感器监听器
    mEventListenerAccelerometer = new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            float[] values = event.values;
            mTextViewAccelerometer.setText(getResources().getString
                (R.string.Accelerometer) + values[0] + "," + values[1] + "," + values[2]);
        }
        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }
    };
    //创建线性加速器监听器
    mEventListenerLinearAcceleration = new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            float[] values = event.values;
            mTextViewLinearAcceleration.setText(getResources().getString
                (R.string.LinearAcceleration) + values[0] + "," + values[1] + "," + values[2]);
        }
        @Override
```

```
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
        //创建重力传感器监听器
        mEventListenerGravity = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;
                mTextViewGravity.setText(getResources().getString
                    (R.string.Gravity) + values[0] + ", " + values[1] + ", " + values[2]);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
        //创建电磁场传感器监听器
        mEventListenerMagneticField = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;
                mTextViewMagneticField.setText(getResources().getString
                    (R.string.Magnetic_Field) + values[0] + "," + values[1] + "," + values[2]);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
        //创建方向传感器监听器
        mEventListenerOrientation = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;
                mTextViewOrientation.setText(getResources().getString
                    (R.string.Orientation) + values[0] + "," + values[1] + "," + values[2]);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
        //创建温度传感器监听器
        mEventListenerTemperature = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;
                mTextViewTemperature.setText(getResources().getString
                    (R.string.Temperature) + values[0]);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
        //创建环境光传感器监听器
        mEventListenerLight = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
```

```
                float[] values = event.values;
                mTextViewLight.setText(getResources().getString(R.string.Light) + values[0]);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
        //创建压力传感器监听器
        mEventListenerPressure = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;
                mTextViewPressure.setText(getResources().getString(R.string.Pressure) + values[0]);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
        //创建旋转向量监听器
        mEventListenerRotationVector = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;
                mTextViewRotationVector.setText(getResources().getString
                    (R.string.Rotation_Vector) + values[0] + "," + values[1] + "," + values[2]);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
        //创建条形码传感器监听器
        mEventListenerBarcode = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
               mTextViewBarcode.setText(getResources().getString(R.string.Barcode)+event.barcode);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };
    }

    @Override
    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(mEventListenerAccelerometer,
                mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                SensorManager.SENSOR_DELAY_FASTEST);          //取加速度传感器信息
        mSensorManager.registerListener(mEventListenerLinearAcceleration,
                mSensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION),
                SensorManager.SENSOR_DELAY_FASTEST);          //取线性加速器信息
        mSensorManager.registerListener(mEventListenerGravity,
                mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY),
                SensorManager.SENSOR_DELAY_FASTEST);          //取重力传感器信息
        mSensorManager.registerListener(mEventListenerMagneticField,
```

```
193                 mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
194                 SensorManager.SENSOR_DELAY_FASTEST);           //取电磁场传感器信息
195         mSensorManager.registerListener(mEventListenerOrientation,
196                 mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
197                 SensorManager.SENSOR_DELAY_FASTEST);           //取方向传感器信息
198         mSensorManager.registerListener(mEventListenerTemperature,
199                 mSensorManager.getDefaultSensor(Sensor.TYPE_TEMPERATURE),
200                 SensorManager.SENSOR_DELAY_FASTEST);           //取温度传感器信息
201         mSensorManager.registerListener(mEventListenerLight,
202                 mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT),
203                 SensorManager.SENSOR_DELAY_FASTEST);           //取环境光线传感器信息
204         mSensorManager.registerListener(mEventListenerPressure,
205                 mSensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE),
206                 SensorManager.SENSOR_DELAY_FASTEST);           //取压力传感器信息
207         mSensorManager.registerListener(mEventListenerBarcode,
208                 mSensorManager.getDefaultSensor(Sensor.TYPE_BARCODE_READER),
209                 SensorManager.SENSOR_DELAY_FASTEST);           //取条形码传感器信息
210         mSensorManager.registerListener(mEventListenerRotationVector,
211                 mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR),
212                 SensorManager.SENSOR_DELAY_FASTEST);           //取旋转向量信息
213     }
214
215     @Override
216     protected void onStop() {
217         mSensorManager.unregisterListener(mEventListenerAccelerometer);
218         mSensorManager.unregisterListener(mEventListenerLinearAcceleration);
219         mSensorManager.unregisterListener(mEventListenerGravity);
220         mSensorManager.unregisterListener(mEventListenerMagneticField);
221         mSensorManager.unregisterListener(mEventListenerOrientation);
222         mSensorManager.unregisterListener(mEventListenerTemperature);
223         mSensorManager.unregisterListener(mEventListenerLight);
224         mSensorManager.unregisterListener(mEventListenerPressure);
225         mSensorManager.unregisterListener(mEventListenerRotationVector);
226         mSensorManager.unregisterListener(mEventListenerBarcode);
227         super.onStop();
228     }
229 }
```

① 第 55、56 行创建一个 SensorManagerSimulator 对象，并连接到 SensorSimulator 模拟器上。

② 第 61～178 行，为 10 个传感器创建传感器监听器，在监听器中重写 onSensorChanged()和 onAccuracyChanged()方法。其中，float[] values = event.values 是从传感器中获得传感数据，有些传感器只有一个数据，有些传感器有三个数据，所以需要使用一个数组来表示。

③ 第 181～213 行，在 onResume()方法中为这 10 个传感器进行注册。

④ 第 216～228 行，在 onStop()方法中注销这 10 个传感器。

(6) 添加权限。打开根目录下的 AndroidManifest.xml 文件，并在<manifest>标签中加入权限：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

【运行结果】 在书中，SensorSimulator 软件所在文件夹路径为"E:\sensorsimulator-2.0-rc1"，那么要首先运行 E:\sensorsimulator-2.0-rc1\bin 下的 sensorsimulator-2.0-rc1.jar

文件。

然后在 Eclipse 中启动 Android 模拟器,然后运行 SensorDemo 项目。

在 SensorSimulator 工具软件窗口的左上角,用鼠标模拟操纵手机,如图 11-23 所示。于是可在 Android 模拟器中看到相应的数据变化,如图 11-24 所示。

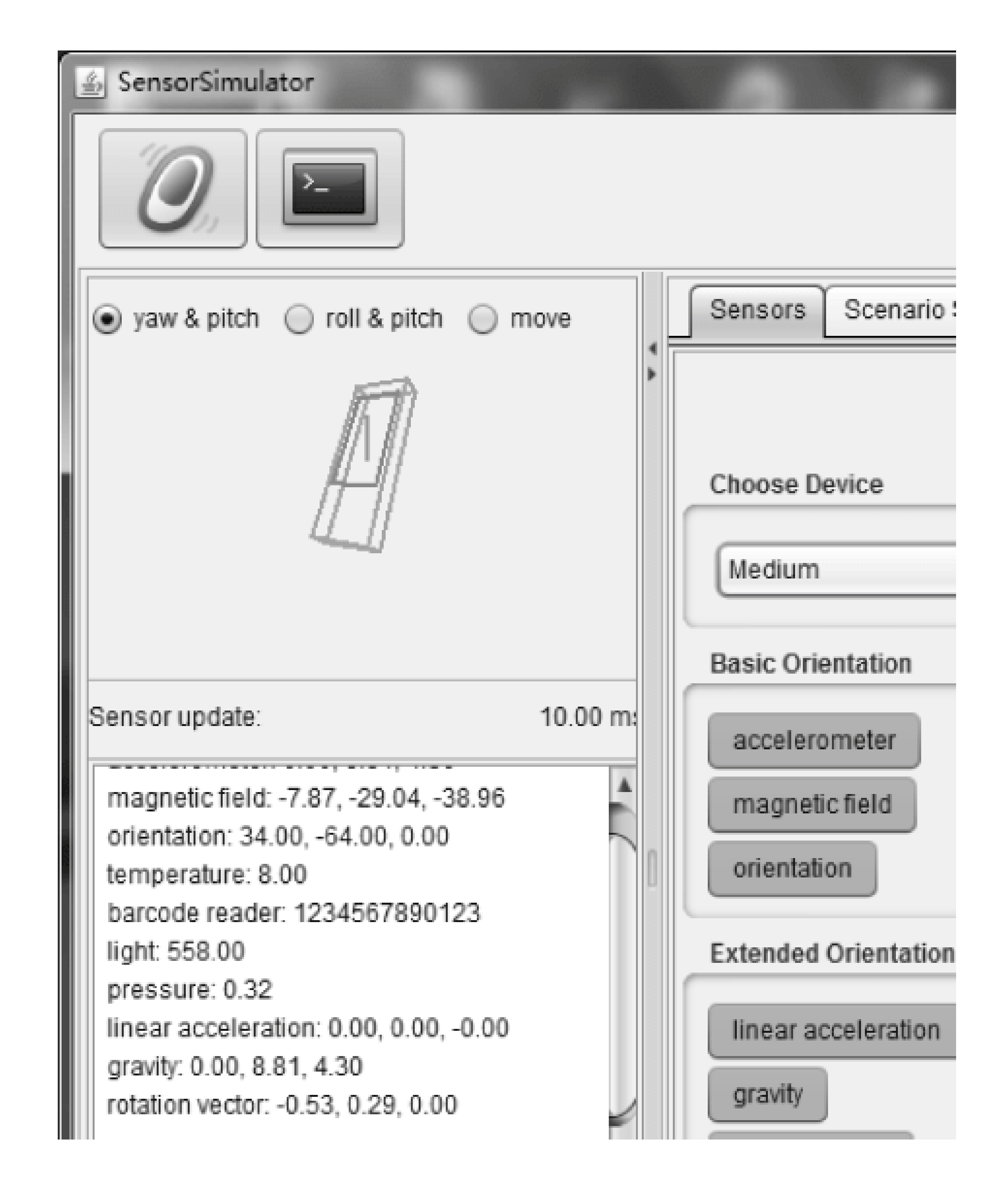






图 11-23　SensorSimulator 模拟的当前数据

图 11-24　Android 模拟器运行的结果

更多的时候,开发一个传感器应用,不仅需要有这些数据,还需要有相应的 UI 界面来体现传感的效果。下面通过一个自定义界面控件的案例来说明这方面的应用。

【案例 11.8】 全方位的圆形水平仪。只感应方向传感器。

【说明】 案例使用自定义的控件来布局水平仪用户界面,这个自定义控件的初始界面将由一个 Java 代码文件来定制。

在代码文件中,以注释的方式给出了在真机上运行时的代码片断。

【开发步骤及解析】

(1) 创建项目。在 Eclipse 中创建一个名为 SensorLevel 的 Android 项目。其应用程序名为 SensorLevel,包名为 cn.com.sgmsc.level,Activity 组件名为 SensorLevelActivity。

(2) 添加 JAR 包。为该项目添加 JAR 包。

(3) 准备图片资源。准备用于水平仪中的标尺条和圆以及位于其上的水泡图片,将它们复制到 res/drawable-mdpi 目录中。

(4) 设计布局。编写 res/layout 目录下的 main.xml 文件。在这里使用 RelativeLayout 布局,代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >

    <cn.com.sgmsc.level.MainView
```

```
8           android:id = "@ + id/mainView"
9           android:layout_width = "fill_parent"
10          android:layout_height  =  "fill_parent"
11          /> <! --  自定义 View -- >
12
13 </LinearLayout >
```

第 7～11 行，声明一个自定义的控件，该控件定义在 cn. com. sgmsc. level 包下，定义的类名为 MainView. java。

(5) 开发自定义控件类代码。在 src/cn. com. sgmsc. level 包下新建一个名为 MainView. java 的文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.level;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;
import android.graphics.Paint.Style;
import android.util.AttributeSet;
import android.view.View;

public class MainView extends View{

 Paint paint  =  new Paint();                              //画笔

 //图片资源的声明

 Bitmap CLB_Bitmap;                                        //中间的大圆图
 Bitmap CLS_Bitmap;                                        //中间的小气泡
 Bitmap HB_Bitmap;                                         //上面的大矩形图
 Bitmap HS_Bitmap;                                         //上面的气泡
 Bitmap VB_Bitmap;                                         //左面的大矩形图
 Bitmap VS_Bitmap;                                         //左面的气泡
 Bitmap HVB_Bitmap;                                        //右下的矩形图
 Bitmap HVS_Bitmap;                                        //右下的气泡

 //背景矩形的位置声明

 int circleB_X  =  70;                                     //中间的大圆图坐标
 int circleB_Y  =  70;
 int horizontalB_X  =  60;                                 //上面的大矩形图坐标
 int horizontalB_Y  =  12;
 int verticalB_X  =  12;                                   //左面的大矩形图坐标
 int verticalB_Y  =  60;
 int HVB_X  =  145;                                        //右下的矩形图坐标
 int HVB_Y  =  145;
 int circleS_X;                                            //中间的小气泡 XY 坐标
 int circleS_Y;
 int horizontalS_X;                                        //上面的气泡 XY 坐标
 int horizontalS_Y;
 int verticalS_X;                                          //左面图的气泡 XY 坐标
```

```
int verticalS_Y;
int HVS_X;                                              //右下的气泡 XY 坐标
int HVS_Y;

public MainView(Context context, AttributeSet attrs){

  super(context,attrs);
  initBitmap();                                         //初始化图片资源
  initLocation();                                       //初始化气泡的位置

}

private void initBitmap(){                              //初始化图片的方法

      CLB_Bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.cl1);
      CLS_Bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.cl2);
      HB_Bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.h1);
      HS_Bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.h2);
      VB_Bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.v1);
      VS_Bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.v2);
      HVB_Bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.hv1);
      HVS_Bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.hv2);
}

private void initLocation(){                            //初始化气泡位置的方法

      circleS_X = circleB_X + CLB_Bitmap.getWidth()/2 - CLS_Bitmap.getWidth()/2;
      circleS_Y = circleB_Y + CLB_Bitmap.getHeight()/2 - CLS_Bitmap.getHeight()/2;
      horizontalS_X = horizontalB_X + HB_Bitmap.getWidth()/2 - HS_Bitmap.getWidth()/2;
      horizontalS_Y = horizontalB_Y + HB_Bitmap.getHeight()/2 - HS_Bitmap.getHeight()/2;
      verticalS_X = verticalB_X + VB_Bitmap.getWidth()/2 - VS_Bitmap.getWidth()/2;
      verticalS_Y = verticalB_Y + VB_Bitmap.getHeight()/2 - VS_Bitmap.getHeight()/2;
      HVS_X = HVB_X + HVB_Bitmap.getWidth()/2 - HVS_Bitmap.getWidth()/2;
      HVS_Y = HVB_Y + HVB_Bitmap.getHeight()/2 - HVS_Bitmap.getHeight()/2;
}

@Override
protected void onDraw(Canvas canvas){                   //重写的绘制方法
    super.onDraw(canvas);
    canvas.drawColor(Color.WHITE);                      //设置背景色为白色
    paint.setColor(Color.BLUE);                         //设置画笔颜色
    paint.setStyle(Style.STROKE);                       //设置画笔为不填充
    canvas.drawRect(5, 5, 315, 315, paint);             //绘制外边框矩形

    //画背景矩形
canvas.drawBitmap(CLB_Bitmap, circleB_X,circleB_Y, paint);        //中
canvas.drawBitmap(HB_Bitmap, horizontalB_X,horizontalB_Y, paint); //上
canvas.drawBitmap(VB_Bitmap, verticalB_X,verticalB_Y, paint);     //左
canvas.drawBitmap(HVB_Bitmap, HVB_X,HVB_Y, paint);                //下

//开始绘制气泡
canvas.drawBitmap(CLS_Bitmap, circleS_X,circleS_Y, paint);        //中
canvas.drawBitmap(HS_Bitmap, horizontalS_X,horizontalS_Y, paint); //上
canvas.drawBitmap(VS_Bitmap, verticalS_X,verticalS_Y, paint);     //左
canvas.drawBitmap(HVS_Bitmap, HVS_X, HVS_Y, paint);               //下
```

```
99  paint.setColor(Color.GRAY);                                  //设置画笔颜色用来绘制刻度
100
101  //绘制上面方框中的刻度
102     canvas.drawLine (horizontalB_X + HB_Bitmap.getWidth()/2 - 7,
103             horizontalB_Y, horizontalB_X + HB_Bitmap.getWidth()/2 - 7,
104             horizontalB_Y + HB_Bitmap.getHeight() - 2, paint);
105  canvas.drawLine (horizontalB_X + HB_Bitmap.getWidth()/2 + 7,
106             horizontalB_Y, horizontalB_X + HB_Bitmap.getWidth()/2 + 7,
107             horizontalB_Y + HB_Bitmap.getHeight() - 2, paint);
108
109  //绘制左面方框中的刻度
110  canvas.drawLine(verticalB_X,verticalB_Y + VB_Bitmap.getHeight()/2 - 7,
111             verticalB_X + VB_Bitmap.getWidth() - 2,
112             verticalB_Y + VB_Bitmap.getHeight()/2 - 7, paint);
113  canvas.drawLine(verticalB_X,verticalB_Y + VB_Bitmap.getHeight()/2 + 7,
114             verticalB_X + VB_Bitmap.getWidth() - 2,
115             verticalB_Y + VB_Bitmap.getHeight()/2 + 7, paint);
116
117  //绘制下面方框中的刻度
118  canvas.drawLine(HVB_X + HVB_Bitmap.getWidth()/2 - 10,
119             HVB_Y + HVB_Bitmap.getHeight()/2 - 20,
120             HVB_X + HVB_Bitmap.getWidth()/2 + 20,
121             HVB_Y + HVB_Bitmap.getHeight()/2 + 10, paint);
122  canvas.drawLine(HVB_X + HVB_Bitmap.getWidth()/2 - 20,
123             HVB_Y + HVB_Bitmap.getHeight()/2 - 10,HVB_X + HVB_Bitmap.getWidth()/2 + 10,
124             HVB_Y + HVB_Bitmap.getHeight()/2 + 20, paint);
125
126  //中间圆圈中的刻度(小圆)
127  RectF oval  =  new RectF(circleB_X + CLB_Bitmap.getWidth()/2 - 10,
128             circleB_Y + CLB_Bitmap.getHeight()/2 - 10,
129             circleB_X + CLB_Bitmap.getWidth()/2 + 10,
130             circleB_Y + CLB_Bitmap.getHeight()/2 + 10);
131  canvas.drawOval(oval, paint);                           //绘制基准线(圆)
132  }
133 }
```

① 第 48～54 行，定义构造方法。在其中使用两个自定义的方法，initBitmap()用于为各图片控件指定图片资源，initLocation()用于计算气泡的初始位置坐标。

② 第 81～132 行，重写 onDraw()方法。其中，第 83～86 行定义画布背景区域，水平仪图片就在该区域内；第 89～92 行绘制水平仪背景图片；第 95～99 行绘制气泡图片；第 102～131 行绘制每一部分的居中刻度线。

(6) 开发逻辑代码。打开 src/cn.com.sgmsc.level 包下的 SensorLevelActivity.java 文件，并编辑之。其代码如下所示。

```
package cn.com.sgmsc.level;

import org.openintents.sensorsimulator.hardware.Sensor;
import org.openintents.sensorsimulator.hardware.SensorEvent;
import org.openintents.sensorsimulator.hardware.SensorEventListener;
import org.openintents.sensorsimulator.hardware.SensorManagerSimulator;

import android.app.Activity;
import android.hardware.SensorManager;
```

```
import android.os.Bundle;

public class SensorLevelActivity extends Activity {      //继承 Activity
    MainView mv;                                         //主 View
    int k = 45;                                          //灵敏度

    //SensorManager mySensorManager;                     //真机时
    SensorManagerSimulator mySensorManager;              //测试时,SensorManager 对象引用
    SensorEventListener myEventSensorListener;

    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                   //设置当前的用户界面
        mv = (MainView) findViewById(R.id.mainView);     //获取主界面
        mySensorManager = SensorManagerSimulator.getSystemService(this,SENSOR_SERVICE);
                                                         //测试时
        mySensorManager.connectSimulator();              //测试时
        //mySensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);  //真机

        myEventSensorListener = new SensorEventListener() {  //传感器监听器类

            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {}
                                                         //重写 onAccuracyChanged()方法

            @Override
            public void onSensorChanged(SensorEvent event) {//重写 onSensorChanged()方法
                float[] values = event.values;
                double pitch = values[SensorManager.DATA_Y];
                double roll = values[SensorManager.DATA_Z];
                int x = 0; int y = 0;          //临时变量,算中间水泡坐标时用
                int tempX = 0; int tempY = 0; //下面气泡的临时变量
                //开始调整 x 的值
                if(Math.abs(roll)<= k){
                    mv.horizontalS_X = mv.horizontalB_X + (int)(((mv.HB_Bitmap.getWidth()
                                 - mv.HS_Bitmap.getWidth())/2.0) - (((mv.HB_Bitmap.getWidth()
                                 - mv.HS_Bitmap.getWidth())/2.0) * roll)/k);  //上面的
                      x = mv.circleB_X + (int)(((mv.CLB_Bitmap.getWidth()
                         - mv.CLS_Bitmap.getWidth())/2.0) - (((mv.CLB_Bitmap.getWidth()
                         - mv.CLS_Bitmap.getWidth())/2.0) * roll)/k);  //中间的
                  }else if(roll > k){
                    mv.horizontalS_X = mv.horizontalB_X;
                    x = mv.circleB_X;
                  }else{
                    mv.horizontalS_X = mv.horizontalB_X + mv.HB_Bitmap.getWidth()
                                   - mv.HS_Bitmap.getWidth();
                    x = mv.circleB_X + mv.CLB_Bitmap.getWidth() - mv.CLS_Bitmap.getWidth();
                }

                //开始调整 y 的值
                if(Math.abs(pitch)<= k){
                     mv.verticalS_Y = mv.verticalB_Y + (int)(((mv.VB_Bitmap.getHeight()
                         - mv.VS_Bitmap.getHeight())/2.0) + (((mv.VB_Bitmap.getHeight()
                         - mv.VS_Bitmap.getHeight())/2.0) * pitch)/k); //左面的
```

```
                y = mv.circleB_Y + (int)(((mv.CLB_Bitmap.getHeight() - mv.CLS_Bitmap.
                getHeight())/2.0)
                + (((mv.CLB_Bitmap.getHeight() - mv.CLS_Bitmap.getHeight())/2.0) *
                pitch)/k);                                    //中间的
            }else if(pitch>k){
                mv.verticalS_Y = mv.verticalB_Y + mv.VB_Bitmap.getHeight() - mv.VS_
                Bitmap.getHeight();
                y = mv.circleB_Y + mv.CLB_Bitmap.getHeight() - mv.CLS_Bitmap.getHeight();
            }else{
                mv.verticalS_Y = mv.verticalB_Y;
                y = mv.circleB_Y;
            }

            //下面的
            tempX = -(int)(((mv.HVB_Bitmap.getWidth()/2 - 28) * roll
                + (mv.HVB_Bitmap.getWidth()/2 - 28) * pitch)/k);
            tempY = -(int)((-(mv.HVB_Bitmap.getWidth()/2 - 28) * roll
                - (mv.HVB_Bitmap.getWidth()/2 - 28) * pitch)/k);

            //限制下面的气泡范围
            if(tempY > mv.HVB_Bitmap.getHeight()/2 - 28){
                tempY = mv.HVB_Bitmap.getHeight()/2 - 28;
            }
            if(tempY < -mv.HVB_Bitmap.getHeight()/2 + 28){
                  tempY = -mv.HVB_Bitmap.getHeight()/2 + 28;
            }
            if(tempX > mv.HVB_Bitmap.getWidth()/2 - 28){
                tempX = mv.HVB_Bitmap.getWidth()/2 - 28;
            }
            if(tempX < -mv.HVB_Bitmap.getWidth()/2 + 28){
                tempX = -mv.HVB_Bitmap.getWidth()/2 + 28;
            }
            mv.HVS_X = tempX + mv.HVB_X + mv.HVB_Bitmap.getWidth()/2
                    - mv.HVS_Bitmap.getWidth()/2;
            mv.HVS_Y = tempY + mv.HVB_Y + mv.HVB_Bitmap.getHeight()/2
                    - mv.HVS_Bitmap.getWidth()/2;

        //中间的水泡在圆内才改变坐标
            if(isContain(x, y)){
              mv.circleS_X = x;
              mv.circleS_Y = y;
             }

             mv.postInvalidate();                  //重绘 MainView
        }

        public boolean isContain(int x, int y){   //判断点是否在圆内
            int tempx = (int)(x + mv.CLS_Bitmap.getWidth()/2.0);
            int tempy = (int)(y + mv.CLS_Bitmap.getWidth()/2.0);
            int ox = (int)(mv.circleB_X + mv.CLB_Bitmap.getWidth()/2.0);
            int oy = (int)(mv.circleB_X + mv.CLB_Bitmap.getWidth()/2.0);
            if(Math.sqrt((tempx - ox) * (tempx - ox) + (tempy - oy) * (tempy - oy))
```

```
112                                >(mv.CLB_Bitmap.getWidth()/2.0 - mv.CLS_Bitmap.getWidth()/2.0)){
                                                                  //不在圆内
113                         return false;
114                     }else{                                    //在圆内时
115                         return true;
116                     }
117                 }
118         };
119     }
120 
121     @Override
122     protected void onResume() {                               //重写的 onResume()方法
123         mySensorManager.registerListener(                     //注册监听
124             myEventSensorListener,                            //监听器 SensorListener 对象
125             mySensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),//只检查方向的
                                                                                 //变化
126             SensorManager.SENSOR_DELAY_UI                                    //频度
127             );
128         super.onResume();
129     }
130 
131     @Override
132     protected void onPause() {                                             //重写 onPause()方法
133         mySensorManager.unregisterListener(myEventSensorListener);  //取消注册监听器
134         super.onPause();
135     }
136 
137 }
```

① 第 35～104 行重写了 onSensorChanged()方法，用于监听传感器方向的采样值变化，并根据其新的采样值重新绘制各部分气泡的位置，这些气泡都被限定在其各自的背景矩形或圆的范围内。

② 第 103 行通过调用 postInvalidate()方法来重绘 MainView。

③ 第 106～117 行是判断点是否在圆内的方法。

④ 第 122～129 行重定 onResume()方法，在该方法中注册方向监听器。

⑤ 第 132～135 行重写 onPause()方法，在该方法中注销方向监听器。

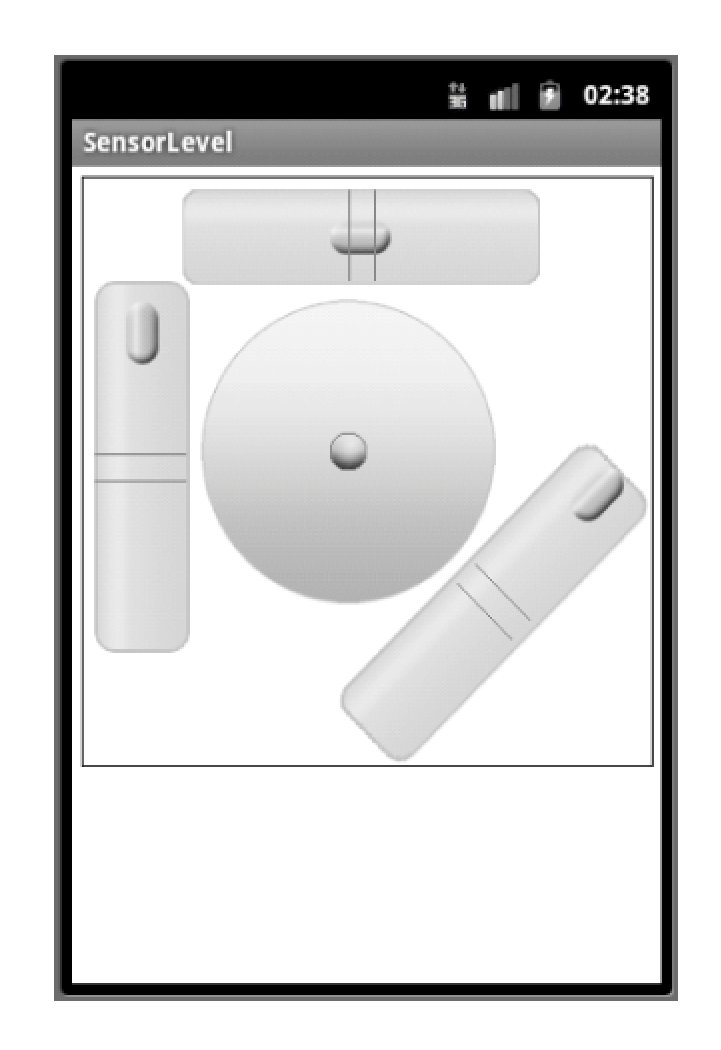
(7) 添加权限。打开根目录下的 AndroidManifest.xml 文件，并在<manifest>标签中加入权限：

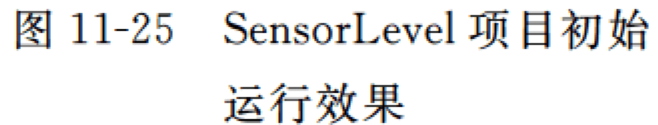
```
<uses-permission android:name=" android.permission.INTERNET"/>
```

【运行结果】 首先运行 E:\ sensorsimulator-2.0-rc1\bin 下的 sensorsimulator-2.0-rc1.jar 文件。

然后在 Eclipse 中启动 Android 模拟器，然后运行 SensorLevel 项目。初始运行效果如图 11-25 所示。

在 SensorSimulator 工具软件窗口的左上角，用鼠标模拟操纵手机，如图 11-26 所示。于是可在 Android 模拟器中看到相应的数据变化，如图 11-27 所示。




图 11-25 SensorLevel 项目初始运行效果

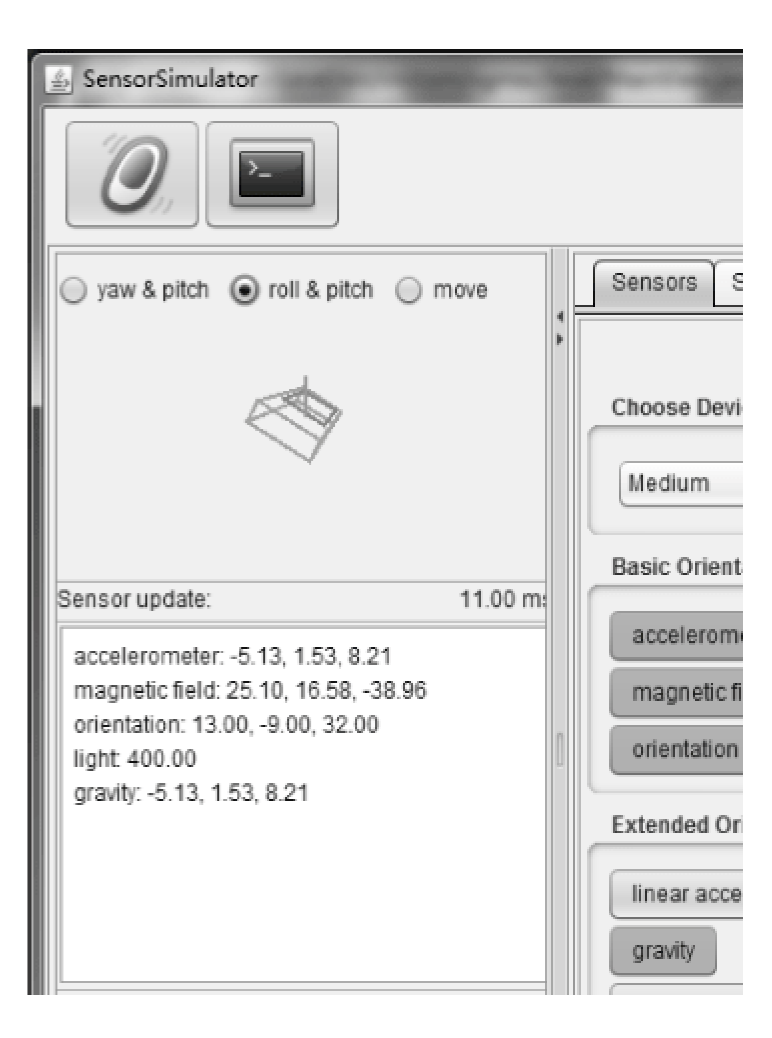


图 11-26 SensorSimulator 模拟的当前数据

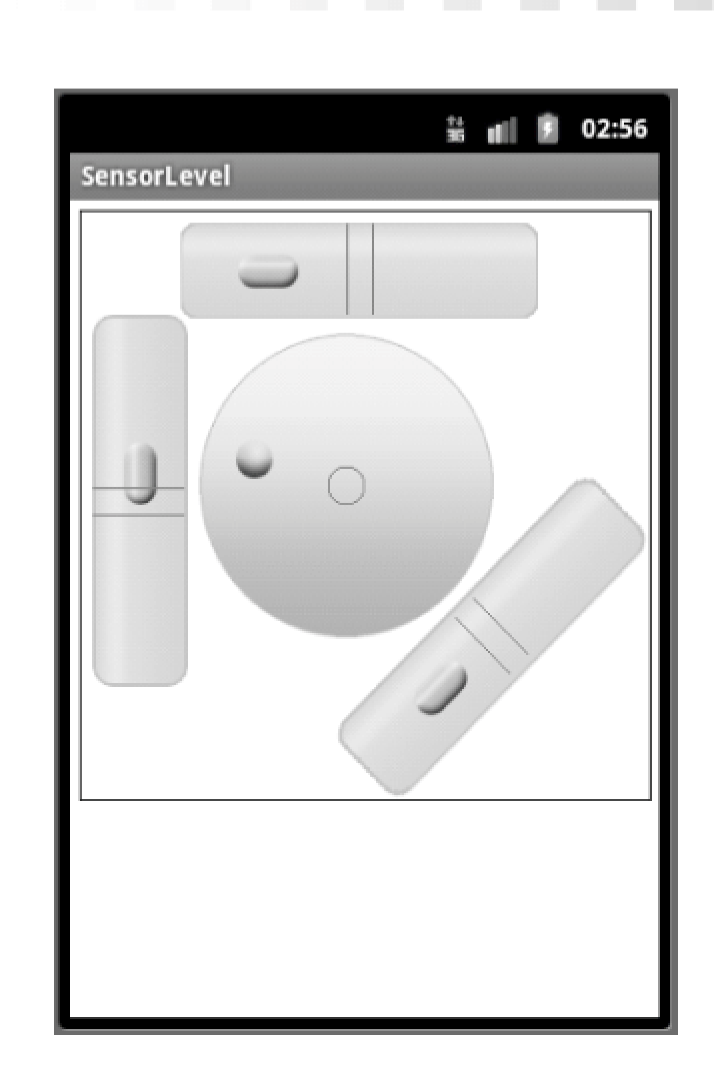


图 11-27 Android 模拟器运行的结果

小结

本章主要介绍了常用的有关手机特性的 Android 应用的开发。对于短信、电话控制，手机系统的设置变化，手机电信网络信息，手机电池电量的监测，以及手机传感器的应用给出了案例进行详细的介绍。在开发手机特性的应用中，有较多的应用需要添加权限声明，请读者要留意。手机传感器涉及手机硬件方面的需求，在模拟器上开发这方面的应用，必须要在 SensorSimulator 工具的协助下进行。

到本章为止，分门别类地介绍了关于 Android 应用开发的基础知识、开发技术和编程技巧，相信读者已具备了 Android 应用项目的开发能力，第 12 章中将进行一个综合实例开发全过程的学习，让读者体会一下完整的应用项目的开发始终。

练习

1. 在“掌上微博”中增加一个收发邮件的功能。
2. 在“掌上微博”中增加方向感应功能。

第12章 应用项目开发实例

手机微博是智能手机继即时通信之外的又一个吸引人的应用。手机随时随地的拍摄功能和随时随地的联网功能，使用手机的微博较之计算机更有优势。运行在手机上的微博是客户端的应用程序，中国几个著名的互联网综合服务提供商如腾讯、新浪、网易等公司都有向用户提供这一应用的下载。当然，也可以根据自己的需要，量身订制自己的手机微博应用。本章将介绍手机微博客户端应用程序开发实例。

12.1 手机微博的功能

随着移动互联网技术的发展，风靡全球的博客也从计算机向移动便携设备发展，并且在风格上趋向于微型化和简捷化。通常，设计一个博客系统包括 Web 服务器、Web 端系统和手机客户端系统。本章主要介绍在 Android 平台下开发的手机客户端的系统设计。

12.1.1 手机微博功能介绍

手机微博客户端主要为用户提供一个信息发布和共享的平台，其功能与 Web 浏览器端系统功能差不多，其总体功能框架如图 12-1 所示。

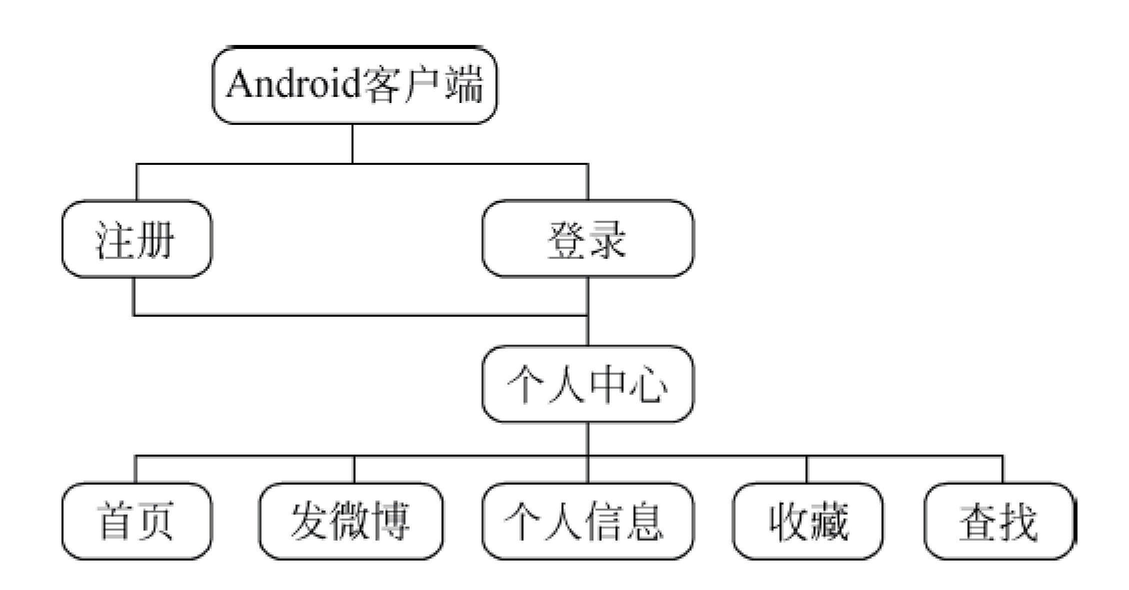


图 12-1 Android 客户端功能框架图

图 12-1 列出了 Android 客户端系统的功能模块，下面对这些模块分别进行简单介绍。

(1) 注册，为初次使用本系统的用户提供注册服务。该模块连接到服务器上，为用户在服务器上申请一个微博用户号，上传头像，记录登录微博的密码、注册时间等信息。

(2) 登录，为已注册的用户使用本系统提供登录窗口。这是进入手机客户端的第一个界面，在登录窗口中提供“登录”、“注册”两个按钮，分别可进入两种不同的状态。

(3) 个人中心，从登录(或注册)窗口进入后就是个人中心。个人中心包含微博客户端系统可以使用的功能和服务，它以选项卡的形式呈现在用户面前，默认情况下显示微博的首页。

(4) 首页,以列表的形式将本用户发送的微博以及本用户所关注的微博全部显示出来,并可以对列表进行分页统计,可以查看各条微博的详细信息以及对各条微博进行转发、评论和收藏。

(5) 发微博,用户在此窗口中可以编辑和发送微博内容,微博文字字符数在150字以内。系统可以实时统计可输入的字符个数。

(6) 个人信息,用户在此窗口查看注册时录入的主要信息,以及本用户已发表的博客数目,关注其他用户数及其关注谁,自己被其他用户关注数(即粉丝数)及其被谁关注等详细信息。

(7) 收藏,管理用户收藏的博客。在收藏窗口中,以列表的方式列出被收藏的所有博客,并可以删除收藏的博客。

(8) 查找,可以模糊搜索其他用户的昵称,并且可以将搜索到的用户添加为自己的好友,即关注该用户。

12.1.2 开发环境和目标平台

1. 开发环境

手机微博客户端是在Android平台下开发的,并且需要访问服务器和数据库。因此开发该手机客户端程序需要用到如下的软件。

(1) Java开发工具,JDK 1.7及其以上版本。本实例使用的JDK版本及安装配置见1.3节描述。

(2) Web应用服务器,Tomcat 7.0及其以上版本。该软件可以从网络上免费下载。本实例使用的Tomcat版本及安装配置见10.2.1节描述。

(3) 数据库,服务器端数据库MySQL Server 5.1及其以上版本和客户端数据库MySQL WorkBench 5.2及其以上版本。这些软件可以从官方网站上免费下载。本书使用的MySQL服务器端安装程序是mysql-essential-5.1.51-win32.exe,客户端安装程序是mysql-workbench-gpl-5.2.28-win32.exe,其安装及配置将在12.2.2节中详细介绍。

(4) 集成开发环境,Eclipse IDE for Java EE Developers。该软件可以从官方网站上免费下载。本书使用的JDK版本及安装配置见1.3节中描述。

(5) Android SDK及其Eclipse开发插件ADT。它们都可以从网络上免费下载。其使用的版本及安装配置见1.3节中描述。

2. 目标平台

手机微博客户端程序开发完成后,经打包签名,可运行在Android 2.1及其以上版本的平台下。

12.2 数据库服务器及Web端应用程序相关说明

本系统在服务器端的MySQL数据库中创建数据库表,Web浏览器客户端和手机客户端将博客的相关信息和用户信息都保存在这个数据库中。在开发系统前,首先要对整个系统的数据库进行设计。

12.2.1 数据库表说明

本系统数据库名为microblog,总共包括8个表,分别为管理员信息表,用户信息表,微博

信息表，收藏信息表，发表评论信息表，回复信息表，转发信息表和关注信息表。下面对其数据表表结构逐一进行介绍，如表 12-1～表 12-8 所示。

表 12-1 管理员信息表 tb_admin

字 段 名	数据类型	字段大小	是否主键	是否可为空	说 明
adm_id	int	11	是	否	管理员 ID 号，主键，自增量。用来唯一标识一组管理员信息的编号
adm_name	varchar	50		否	管理员名称
adm_password	varchar	50		否	管理员密码
adm_lastLoginTime	date				最近一次登录的日期
adm_lastLoginIp	varchar	50			最近一次登录的 IP 地址

表 12-2 用户信息表 tb_user

字 段 名	数据类型	字段大小	是否主键	是否可为空	说 明
u_id	int	11	是	否	用户 ID 号，主键，自增量。用来唯一标识一组用户信息的编号
u_name	varchar	50		否	用户名
u_password	varchar	50		否	密码
u_picture	varchar	50			头像
u_trueName	varchar	20			真实姓名
u_age	int	11			年龄
u_lastLoginTime	date				最近一次的登录时间
u_registerTime	date				注册时间
u_isActivity	bit	1			用户是否有效
u_lastLoginIp	varchar	50			最近一次的登录 IP

表 12-3 微博信息表 tb_microblog

字 段 名	数据类型	字段大小	是否主键	是否可为空	说 明
mb_id	int	11	是	否	微博 ID 号，主键，自增量。用来唯一标识一组微博信息的编号
mb_content	longtext			否	微博内容
mb_time	date				写微博时间
mb_ip	varchar	50			写微博所用 IP
u_id	int	11			用户 ID 号，外键。来自 tb_user 表中

表 12-4 收藏信息表 tb_collection

字 段 名	数据类型	字段大小	是否主键	是否可为空	说 明
cc_id	int	11	是	否	收藏 ID 号，主键，自增量。用来唯一标识一组收藏信息的编号
cc_time	date				收藏的时间
mb_id	int	11			微博 ID 号，外键。来自 tb_microblog 表中
u_id	int	11			用户 ID 号，外键。来自 tb_user 表中

表 12-5　发表评论信息表 tb_comment

字 段 名	数据类型	字段大小	是否主键	是否可为空	说　明
cm_id	int	11	是	否	评论 ID 号，主键，自增量。用来唯一标识一组评论信息的编号
cm_content	longtext			否	评论内容
cm_time	date				评论时间
cm_ip	varchar	50			评论所用 IP
mb_id	int	11			微博 ID 号，外键。来自 tb_microblog 表中
u_id	int	11			用户 ID 号，外键。来自 tb_user 表中

表 12-6　回复信息表 tb_reply

字 段 名	数据类型	字段大小	是否主键	是否可为空	说　明
rp_id	int	11	是	否	回复 ID 号，主键，自增量。用来唯一标识一组回复信息的编号
rp_content	longtext			否	回复内容
rp_time	date				回复时间
rp_ip	varchar	50			回复所用的 IP
cm_id	int	11			评论 ID 号，外键。来自 tb_comment 表中
u_id_from	int	11			发送回复用户 ID 号，外键。来自 tb_user 表中
u_id_to	int	11			接收回复用户 ID 号，外键。来自 tb_user 表中

表 12-7　转发信息表 tb_transmit

字 段 名	数据类型	字段大小	是否主键	是否可为空	说　明
tm_id	int	11	是	否	转发 ID 号，主键，自增量。用来唯一标识一组转发信息的编号
tm_content	longtext			否	转发内容
tm_time	date				转发时间
mb_id	int	11			原微博 ID 号，外键。来自 tb_microblog 表中
u_id	int	11			原微博的用户 ID 号，外键。来自 tb_user 表中

表 12-8　关注信息表 tb_follow

字 段 名	数据类型	字段大小	是否主键	是否可为空	说　明
fw_id	int	11	是	否	关注 ID 号，主键，自增量。用来唯一标识一组关注信息的编号
fw_time	date				关注时间
fans_id	int	11			粉丝(即关注我的用户)的 ID 号
idol_id	int	11			关注用户 ID 号

这些数据库表可以在 MySQL 的 Workbench 中创建，也可以使用 SQL 脚本文件来创建。本实例将使用 mobile-blog.sql 脚本语句文件来创建，具体操作见 12.2.2 节中描述。

12.2.2 MySQL 安装配置和微博系统数据库创建

MySQL 数据库环境搭建分为 MySQL 服务器安装、服务器配置和 MySQL 专用的数据库设计工具 Workbench 的安装三部分。在 MySQL 数据库环境搭建好了之后，就可以利用 mobile-blog.sql 脚本文件来创建数据库。

1. 安装 MySQL 服务器

从网络上下载 MySQL 的安装文件 mysql-essential-5.1.51-win32.exe，并运行此文件，具体安装步骤如下。

(1) 运行 mysql-essential-5.1.51-win32.exe 程序，出现如图 12-2 所示的安装向导。

(2) 单击 Next 按钮，进入下一页向导，在该页中可选择安装类型，有 Typical(典型安装，是默认选项)、Complete(完全安装)、Custom (用户自定义安装)三个选项，这里选择 Typical。

(3) 再单击 Next 按钮，选择安装路径，本实例使用向导默认的安装路径。然后，单击 Install 按钮开始安装。

(4) 在安装向导中，一直单击 Next 按钮，直到完成，如图 12-3 所示。

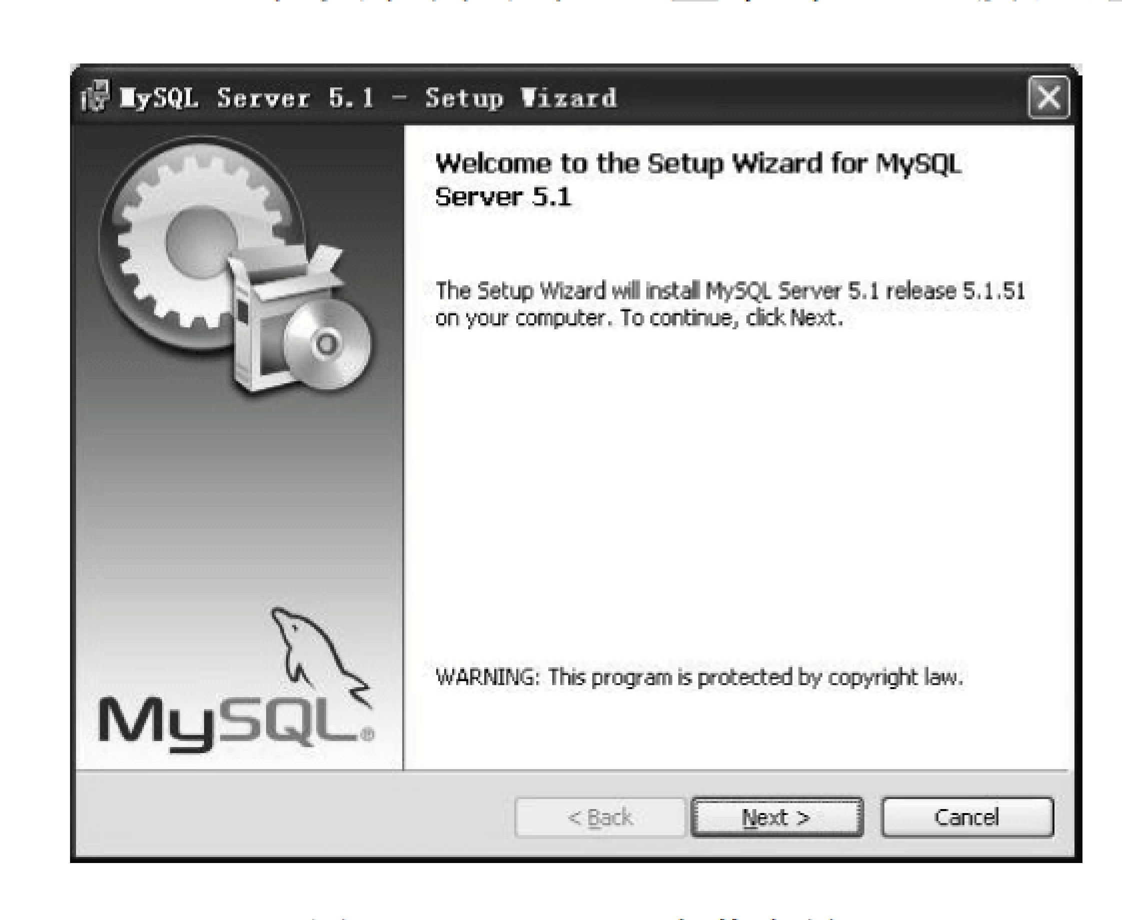


图 12-2 MySQL 安装向导



图 12-3 MySQL 安装向导完成对话框

2. MySQL 服务器配置步骤

在完成 MySQL 服务器安装之后，要根据应用的需要对其进行适当的配置。下面通过图例说明本实例对 MySQL 服务器的配置。

(1) 启动配置向导。如果在图 12-3 中勾选了 Configure the MySQL Server now 项，在单击 Finish 之后会自动进入 MySQL 服务器配置向导。如果没有勾选此项，则在“开始”菜单的“程序”中选择 MySQL→MySQL Server 5.1→MySQL Server Instance Configuration Wizard，进入 MySQL 的服务器配置向导，如图 12-4 所示。然后单击 Next 按钮。

(2) 重新配置实例。如果以前已经安装了 MySQL Server，现在要重新配置时，则出现如图 12-5 所示的向导，在此对话框中选择 Reconfigure Instance 项，然后单击 Next 按钮。如果

是新安装 MySQL Server 不会出现这一步骤。



图 12-4　MySQL 服务器配置向导

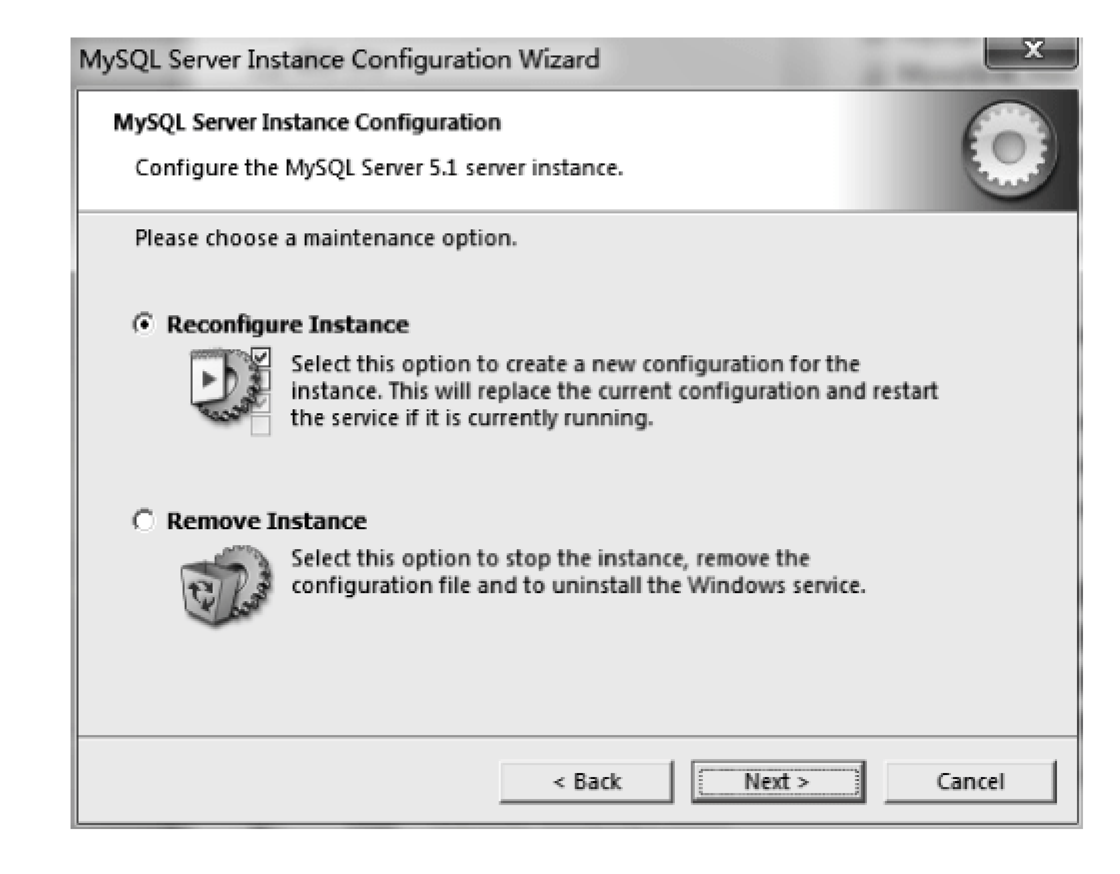


图 12-5　重新配置 MySQL 服务器选择向导

(3) 选择配置类型。在此向导页选择 Detailed Configuration 选项，如图 12-6 所示。然后单击 Next 按钮。

(4) 选择服务器类型。在此向导页选择 Server Machine 选项，使得将 MySQL Server 安装在网络的服务器中，如图 12-7 所示。然后单击 Next 按钮。

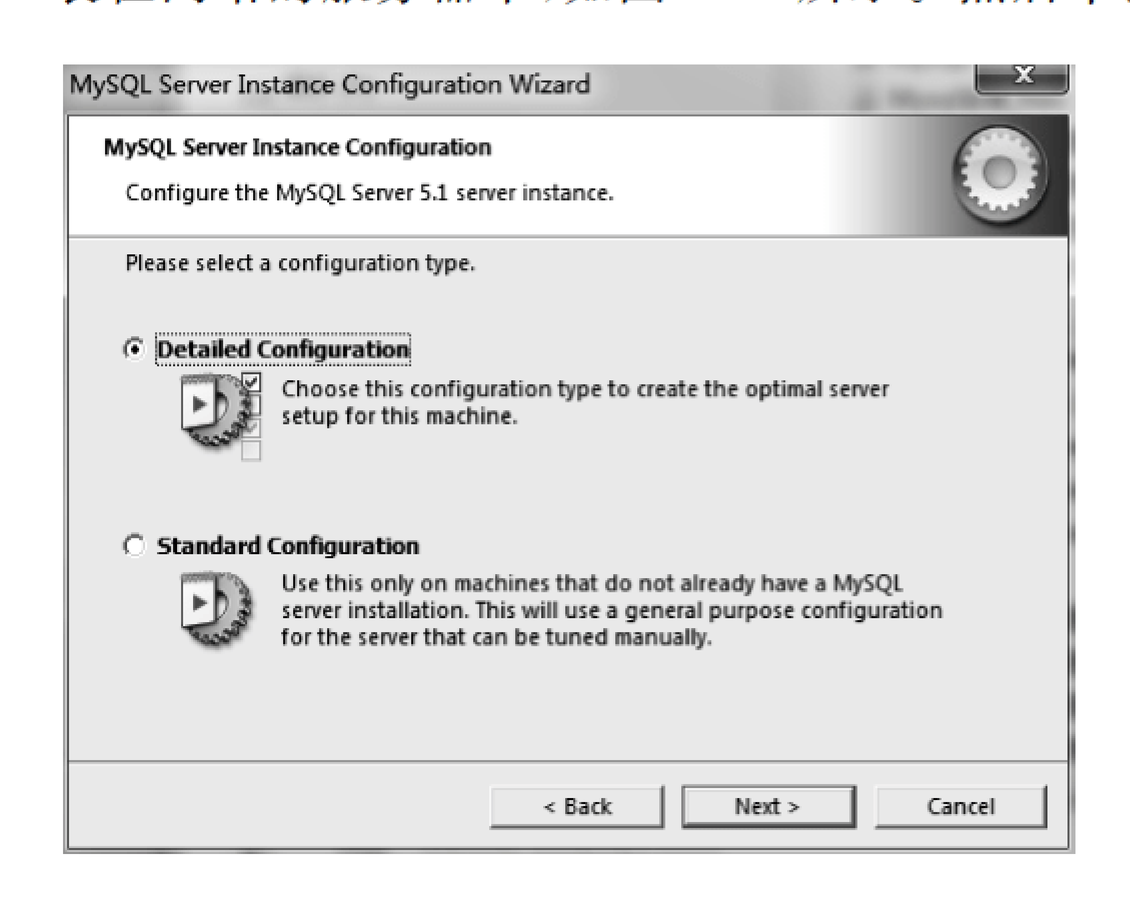


图 12-6　选择配置类型向导页



图 12-7　选择服务器类型向导页

(5) 选择数据库用法。在此向导页选择 Multifunctional Database 选项，如图 12-8 所示。然后单击 Next 按钮，进入下一页，该页将对 InnoDB Tablespace 进行配置，即为 InnoDB 数据库文件选择一个存储空间，本实例不对其作任何修改，继续单击 Next 按钮。

(6) 设置网站的访问量，即可同时连接用户数。在此向导页中选择 Manual Setting 项，如图 12-9 所示。然后单击 Next 按钮。

(7) 设置网络选项。在此向导页中勾选 Enable TCP/IP Networking 和 Enable Strict Mode 选项，使用默认端口号 3306，如图 12-10 所示。然后单击 Next 按钮。

(8) 设置字符集。这一步比较重要，只有正确地设置应用系统的字符集，在运行时才能正常显示汉字。在此向导页中选择 Manual Selected Default Character Set/Collation 选项，并且在 Character Set 中选择 utf8，如图 12-11 所示。然后单击 Next 按钮。



图 12-8 选择数据库用法向导页

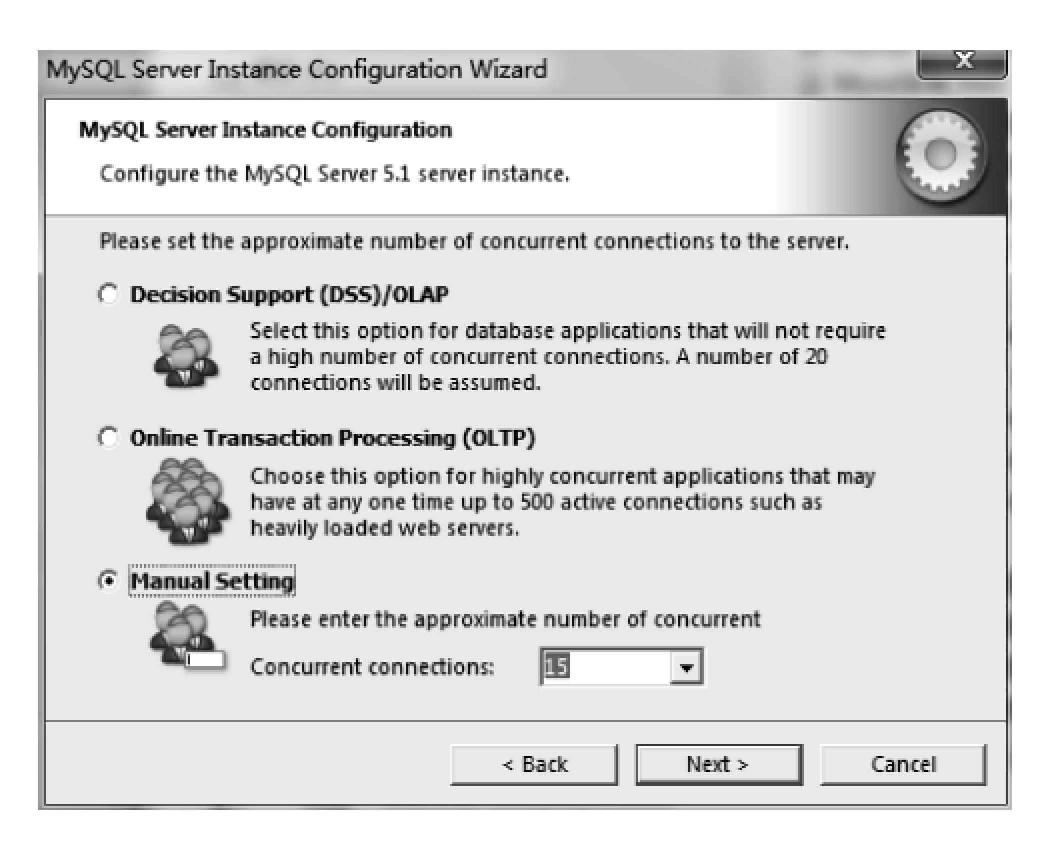


图 12-9 选择可连接用户数向导页

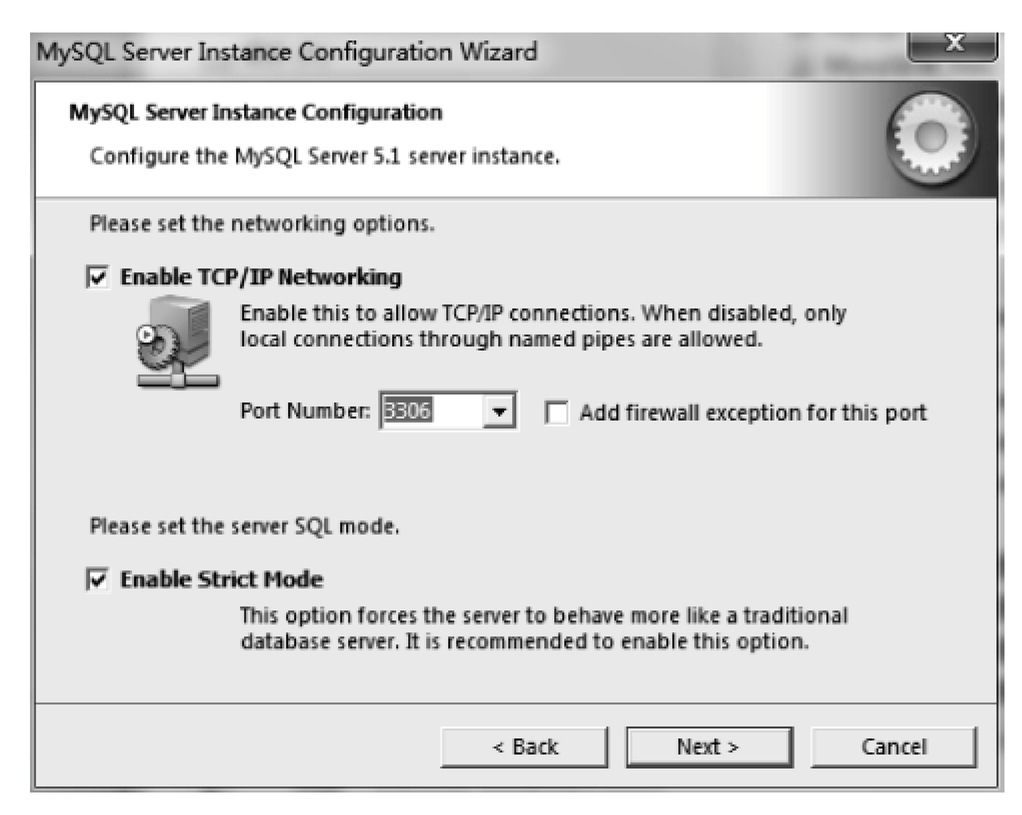


图 12-10 设置网络选项向导页

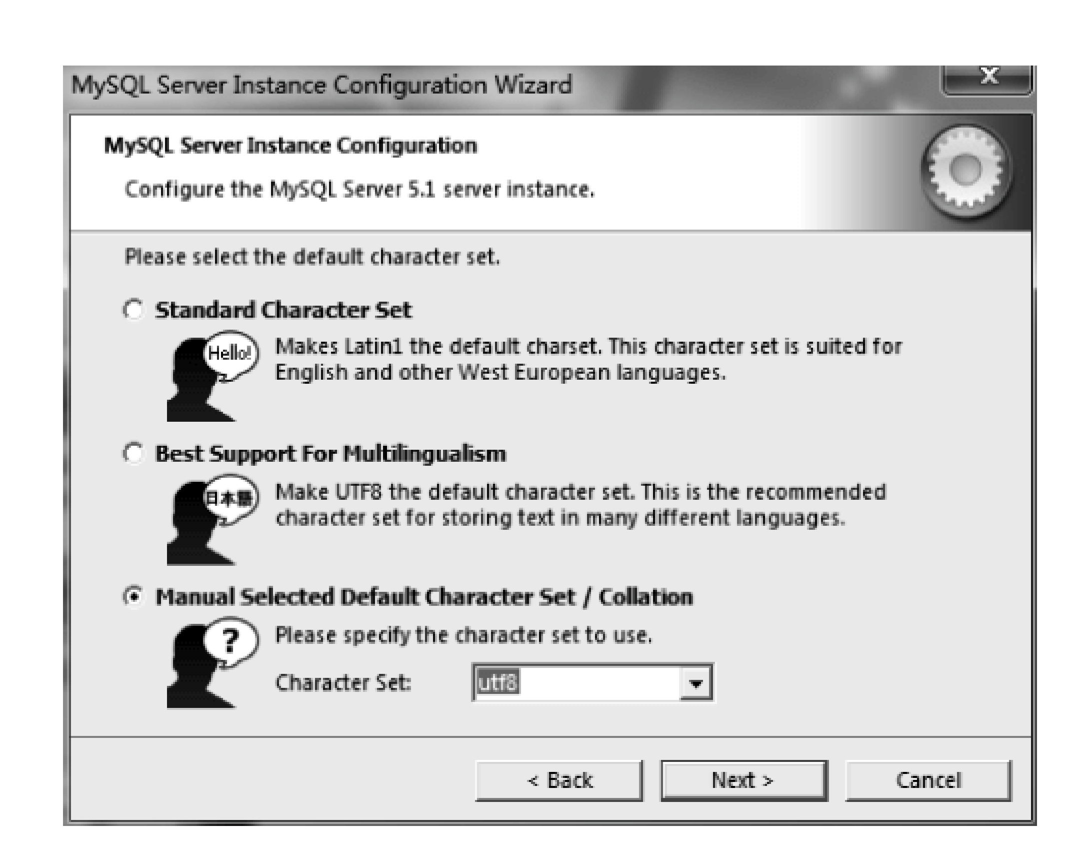


图 12-11 设置字符集向导页

(9) 设置 MySQL 在 Windows 操作系统中的选项。勾选 Include Bin Directory in Windows PATH 项，将 MySQL 的安装文件夹路径添加到 Windows 的环境参数 path 中，如图 12-12 所示。然后单击 Next 按钮。

(10) 设置 MySQL 的安全选项。如果是重新配置 MySQL，则勾选 Modigy Security Settings 项，可以修改密码，如图 12-13 所示。如果是全新安装 MySQL，则只出现 New root password 和 Confirm(输入 root 密码和确认密码)两个编辑框。如果不设置密码，则留空即可。在此页，输入密码为"root"，并勾选 Enable root access from remote machines。然后单击 Next 按钮。



图 12-12 设置 Windows 选项向导页



图 12-13 设置安全选项向导页

(11) 执行配置。在完成上述设置后,在此向导页单击 Execute 按钮,将逐项进行执行设置,执行通过则在此小项前打钩,执行不通过则打叉,如图 12-14 所示。当所有项都打钩后即配置成功,单击 Finish 按钮完成配置。

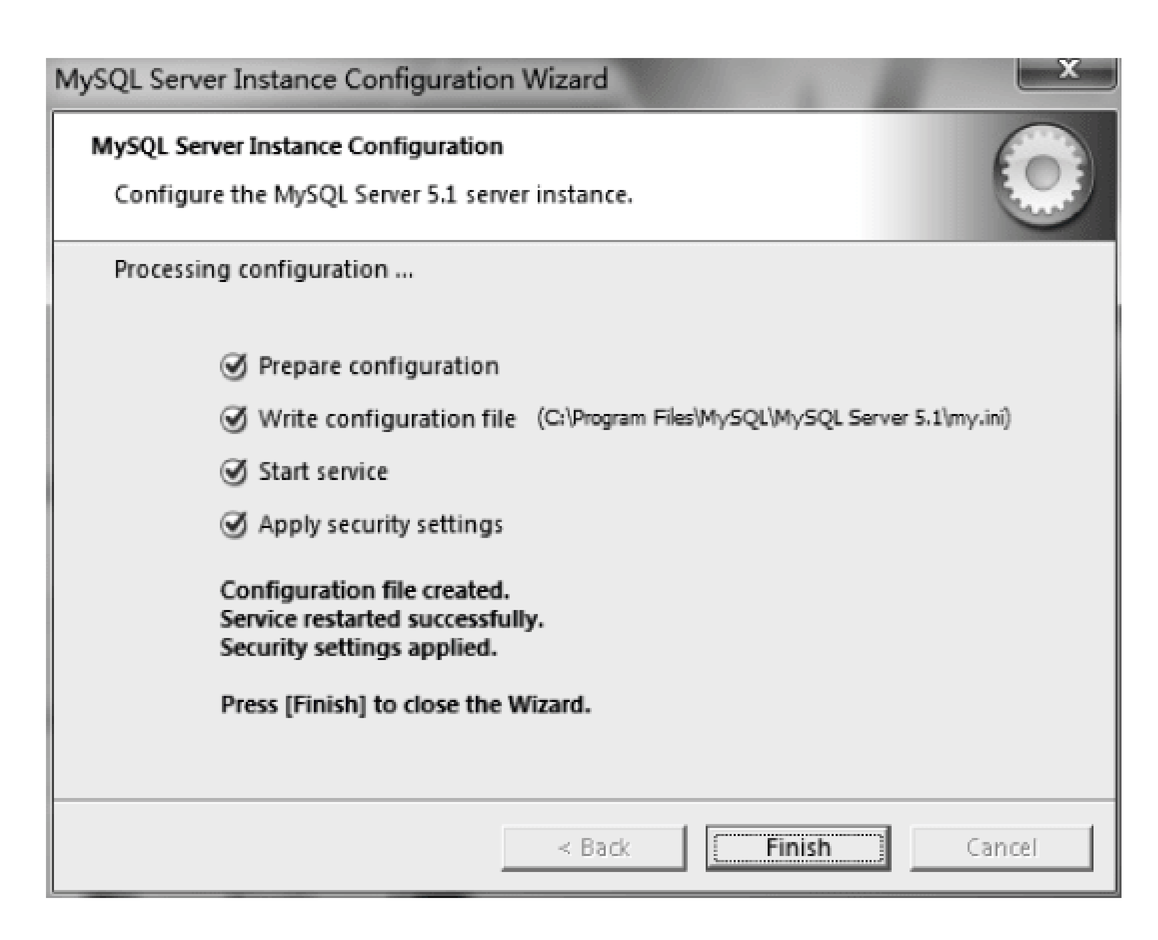


图 12-14　完成配置向导页

在 MySQL 服务器的配置中,比较常见的错误是不能"Start service",出现这个错误通常是以前有安装 MySQL 服务器的情形。解决的办法是,首先检查是否按前面的步骤进行配置;其次是确定之前的密码是否有修改;如果不能解决问题,可以将 MySQL 安装文件夹下的 data 文件夹备份(如果需要保留以前的数据库表),将以前安装的 MySQL 服务器彻底卸载掉,重新安装 MySQL,再将备份的 data 文件夹移回来,然后再重启 MySQL 服务就可以了,这种情况下,可能需要将数据库检查一下,然后修复一次,防止数据出错。

3. 安装数据库设计工具 MySQL Workbench

MySQL Workbench 是 MySQL AB 发布的可视数据库设计工具。这个工具是设计 MySQL 数据库的专用工具,它拥有很多的功能和特性。

安装 MySQL Workbench,只需运行 mysql-workbench-gpl-5.2.28-win32.exe 程序,然后按照安装向导逐步进行即可。

4. 导入数据

创建数据库表的方式通常有两种。一种是通过 MySQL Workbench 工具创建并录入数据,另一种是通过已有的 SQL 脚本文件进行生成。本书为大家提供了一个 SQL 脚本文件,因此可使用第二种方式进行数据库表的创建并导入数据。在本书的源代码文件夹 Ch12 中存有生成本实例数据库表的 SQL 脚本文件 mobile-blog.sql。注意,在 MySQL Workbench 中,利用 SQL 脚本创建数据库表时,其脚本文件名及其存放路径不能包含中文名。

下面介绍创建数据库及导入数据操作过程。

(1) 启动 MySQL Server。注意,在启动 MySQL Workbench 之前必须先启动 MySQL Server。方法是在"开始"菜单的"程序"中选择 MySQL→MySQL Server 5.1→MySQL Command Line Client,进入命令行状态,输入 MySQL Server 的进入密码(本实例密码为 root),出现如图 12-15 所示的信息表示 MySQL Server 启动完成。

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.51-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

图 12-15　进入 MySQL Server 命令行界面

（2）启动 MySQL Workbench。方法是在“开始”菜单的“程序”中选择 MySQL→MySQL Workbench 5.2 CE，即可进入 MySQL Workbench 的欢迎界面，如图 12-16 所示。在欢迎界面停留片刻后，自动进入 MySQL Workbench 工作界面，如图 12-17 所示。



图 12-16　MySQL Workbench 的欢迎界面

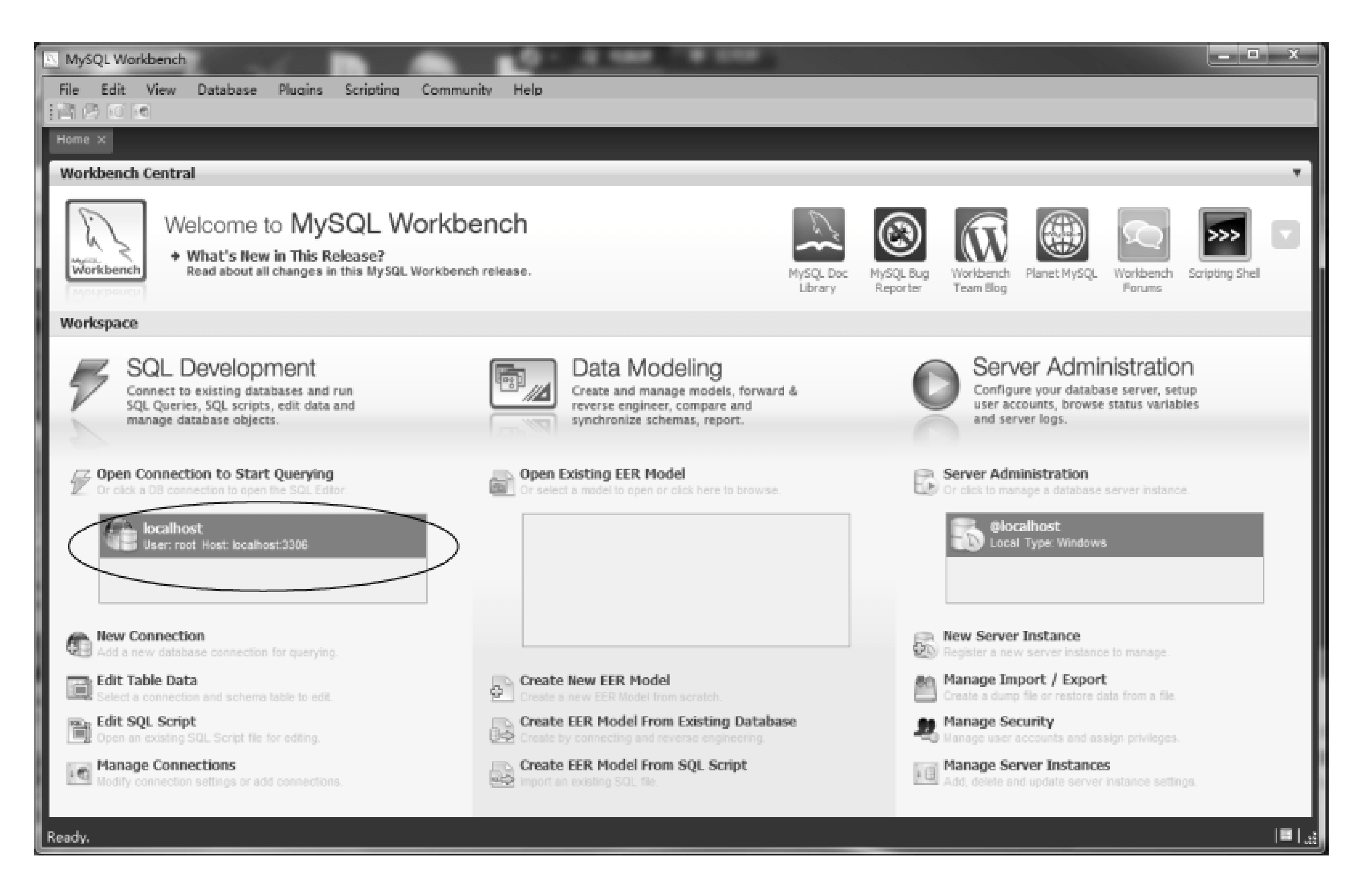


图 12-17　MySQL Workbench 的工作界面

（3）进入 SQL Development。双击“localhost/User：root Host：localhost：3306”（如图 12-17 所示标记处），进入 MySQL Workbench 数据库表管理窗口。

(4) 选择菜单 File→Open SQL Script…,查找文件 mobile-blog.sql 并打开,如图 12-18 所示。

mobile-blog.sql ×

```
/*
SQLyog Community v8.8 Beta2
MySQL - 5.1.51-community : Database - microblog
*********************************************************************
*/

/*!40101 SET NAMES utf8 */;

/*!40101 SET SQL_MODE=''*/;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
CREATE DATABASE  IF NOT EXISTS `microblog` DEFAULT CHARACTER SET utf8;

USE `microblog`;

/*Table structure for table `tb_admin` */

DROP TABLE IF EXISTS `tb_admin`;

CREATE TABLE `tb_admin` (
  `adm_id` int(11) NOT NULL AUTO_INCREMENT,
  `adm_name` varchar(50) NOT NULL,
  `adm_password` varchar(50) NOT NULL,
  `adm_lastLoginTime` date DEFAULT NULL,
  `adm_lastLoginIp` varchar(50) DEFAULT NULL,
```

图 12-18　mobile-blog.sql 文件内容

(5) 执行 mobile-blog.sql 文件。在 MySQL Workbench 的工具栏中单击“执行 SQL 脚本工具”按钮,如图 12-19 所示。

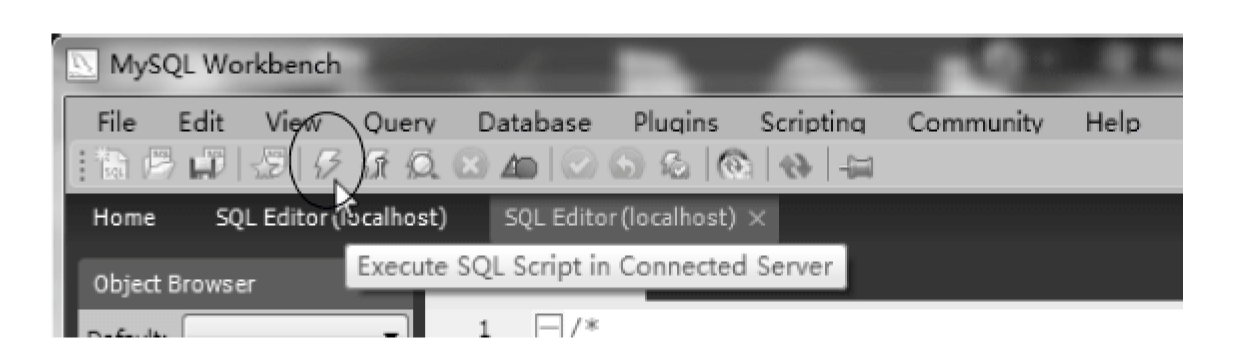


图 12-19　MySQL Workbench 工具栏

执行完成后即可在数据库中看到新创建的数据库 microblog,如图 12-20 所示。

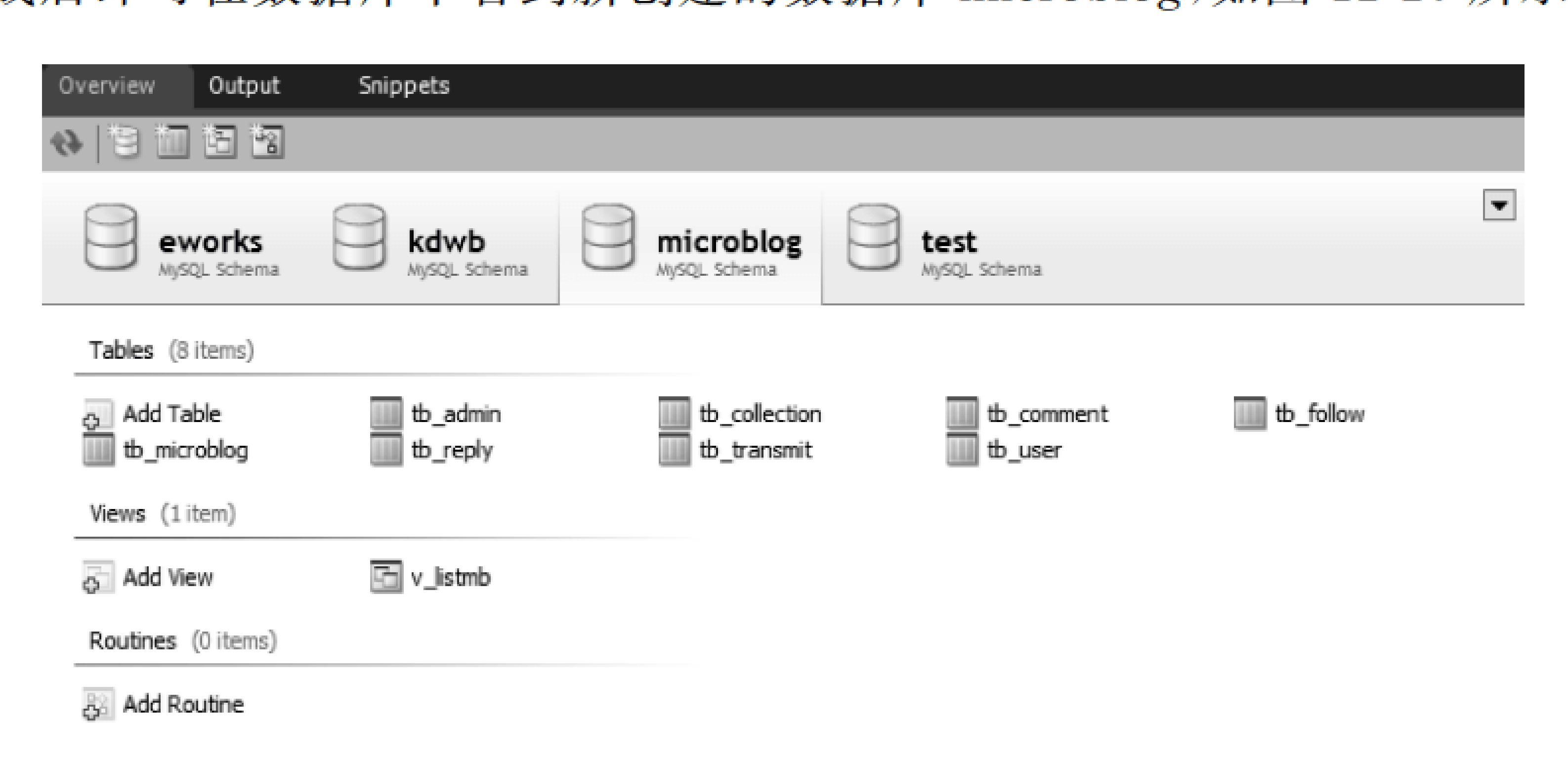


图 12-20　microblog 数据库

在 mobile-blog.sql 脚本文件中,既包括 Create Database 和 Create Table 命令创建数据库表,也包括 Insert 命令向数据表中插入了部分数据。所以在执行 mobile-blog.sql 之后,既创建了数据库表,也完成了数据的导入。可以打开其中的数据表,如 tb_microblog 表,可以看到表中的数据,如图 12-21 所示。



图 12-21 tb_microblog 表中数据

12.2.3 Web 应用服务器的配置和部署

在作为 Web 应用服务器的计算机上安装 Tomcat。本实例中，将开发机作为 Web 应用服务器，在本机上安装 Tomcat 服务器，Tomcat 的安装步骤及其在 Eclipse 中的设置参见 10.2.1 节。接下来进行应用服务器配置和部署

1. 数据源配置

为了使得 Tomcat 服务器与 MySQL 数据库进行连接，Tomcat 中必须有 MySQL 的数据库驱动。添加 MySQL 的数据库驱动非常简单，只需要向 Tomcat 安装文件夹的 lib 子文件夹下拷入 JAR 包 mysql-connector-java-5.1.13-bin.jar 即可，该 JAR 包可以从网络上免费下载获得。

2. Web 应用程序部署

本章着重介绍手机客户端程序的开发，所以不对 Web 浏览器端的程序作详细介绍，只是直接将 Web 浏览器端的应用程序压缩文件包 MobileBlog.war 复制到 Tomcat 安装文件夹的 webapps 文件夹下，该文件可从本书的源代码文件夹的 Ch12 子文件夹中得到。这里的.war 格式文件是 Java 的归档文件，它用于封装 Web 程序模块。

本书的 Tomcat 的安装文件夹路径为 E:\apache-tomcat-7.0.11，所以应该将 MobileBlog.war 文件复制到 E:\apache-tomcat-7.0.11\webapps 文件夹下即可。

12.3 手机客户端的编程实现

根据 9.5 节介绍的应用开发步骤，逐步进行 12.1.1 节中功能模块的实现。

12.3.1 用户界面设计及资源准备

首先，根据本应用项目的功能规划出需要的用户界面数目、界面完成的功能、相互的跳转关系，使得开发人员对整个应用系统有个概要理解。

其次，对每一个用户界面进行初步设计：画出草图，从而得到每个界面所需要的控件、颜色、文字串、图片、图标以及音乐等相关的资源。

再次，根据每个界面的需要，准备相关的文字、图片、颜色、音乐等资源。

上述这些都是在进入详细开发前的准备工作，开发人员必须完成这一步，才能进入后续的代码编写工作。

12.3.2 应用项目

在 Eclipse 中新建一个 Android Application 项目 MobileBlog_Android，然后进行开发。本书中提供了已完成的项目源码，可从本书源代码的 Ch12 文件夹中获得，然后通过 Import 将其载入 Eclipse 中。

1. Import 项目的操作步骤

(1) 在 Eclipse 中，选择菜单 File→Import...，打开 Import 对话框，展开 General，选择 Existing Projects into Workspace 项，如图 12-22 所示。然后单击 Next 按钮。

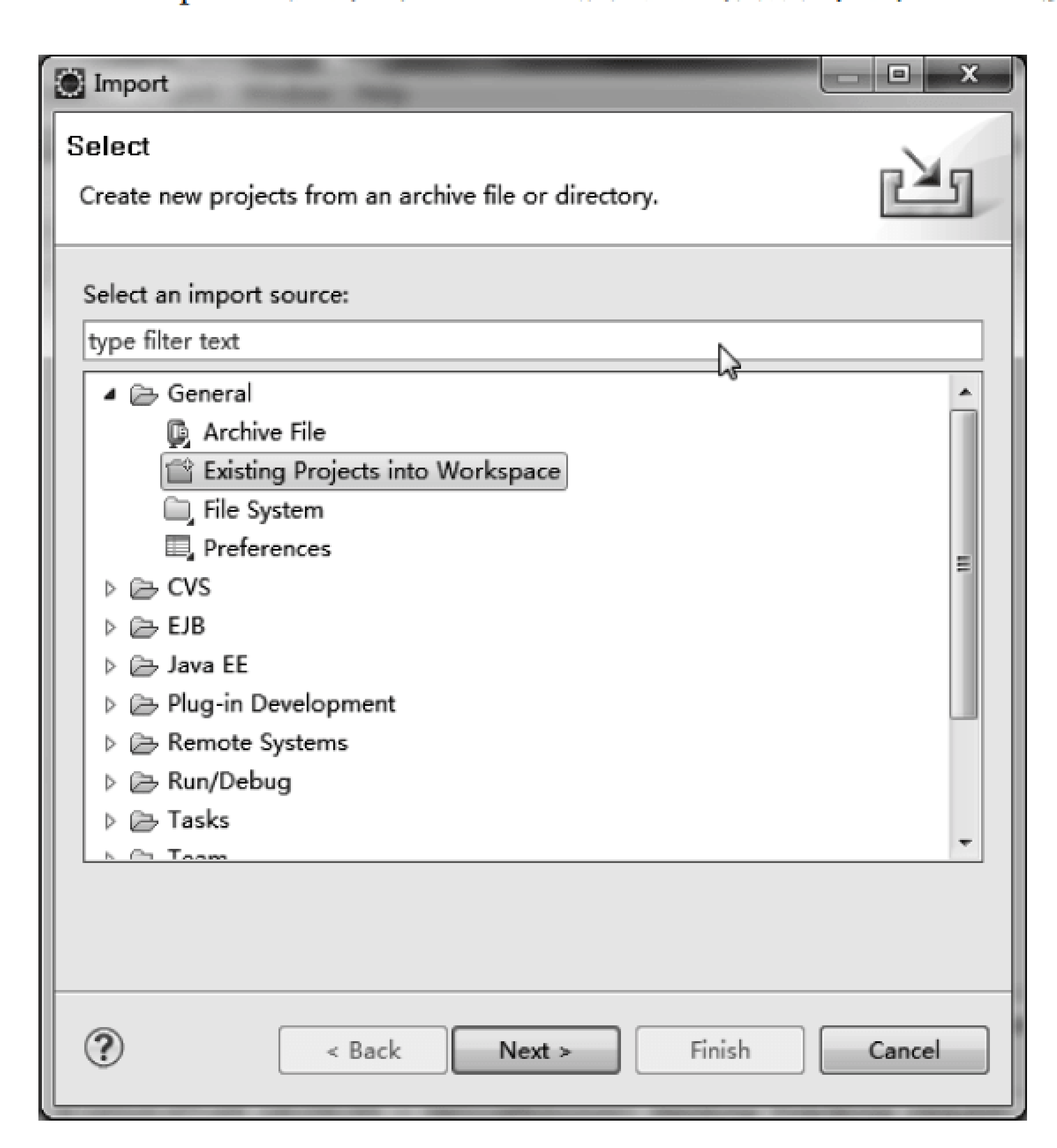


图 12-22 Import 对话框

(2) 在 Import Projects 页中查找到项目的存放位置，本书存放于 E:\AndroidCode\Ch12 下，如图 12-23 所示。然后单击 Finish 按钮，即可将本书提供的应用项目加载到 Eclipse 的当前 Workspace 中。

2. 应用项目的目录结构

在 Eclipse 的 Package Explorer 中应该有 MobileBlog_Android 和 Servers 两个项目，如图 12-24 所示，手机微博客户端才能正常运行，其中 Servers 项目是由配置 Tomcat 服务器和部署微博系统的 Web 应用程序时生成的。在此，只关心 MobileBlog_Android 项目，它是手机微博客户端的程序项目。

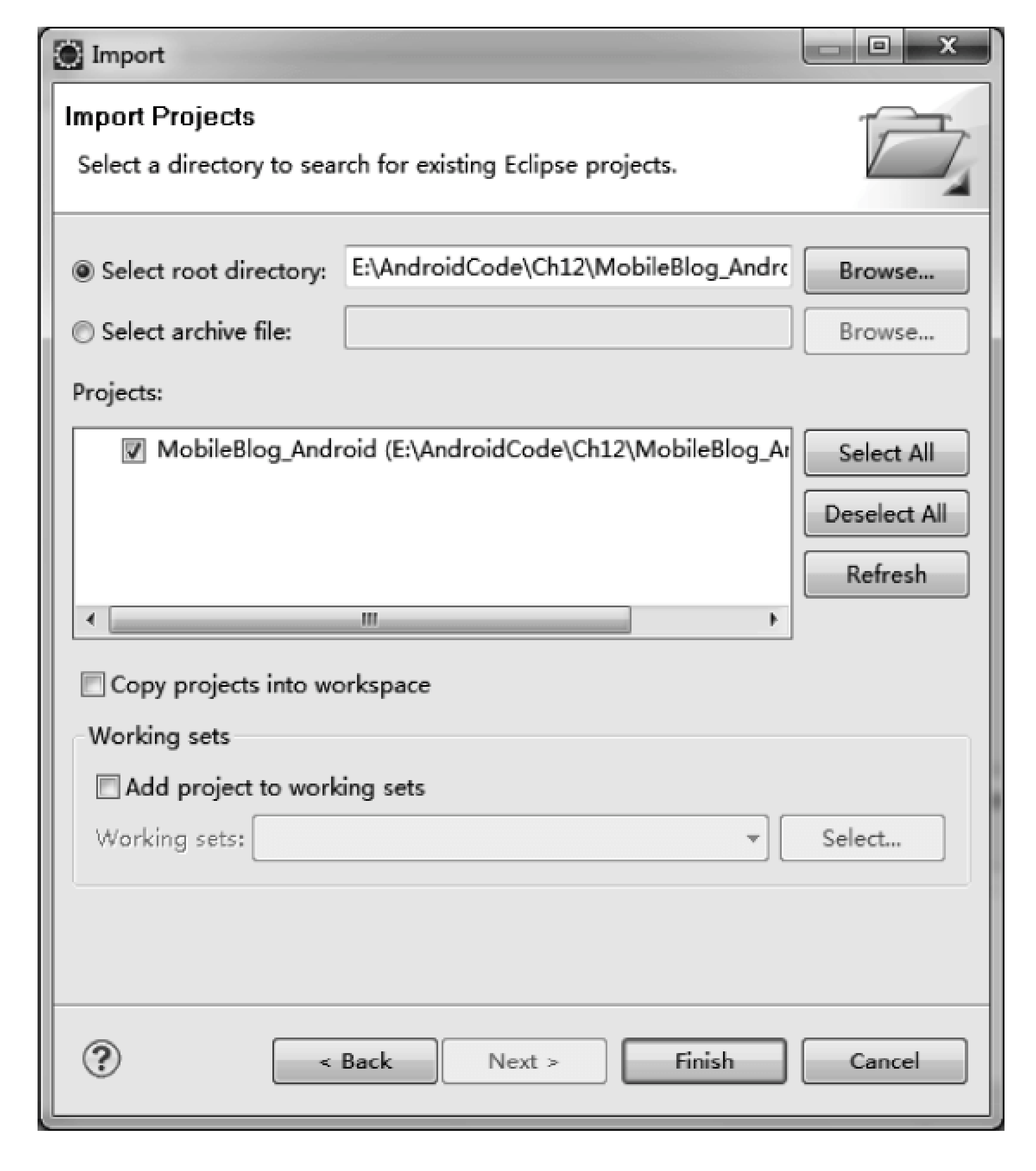


图 12-23　选择要载入的项目文件夹

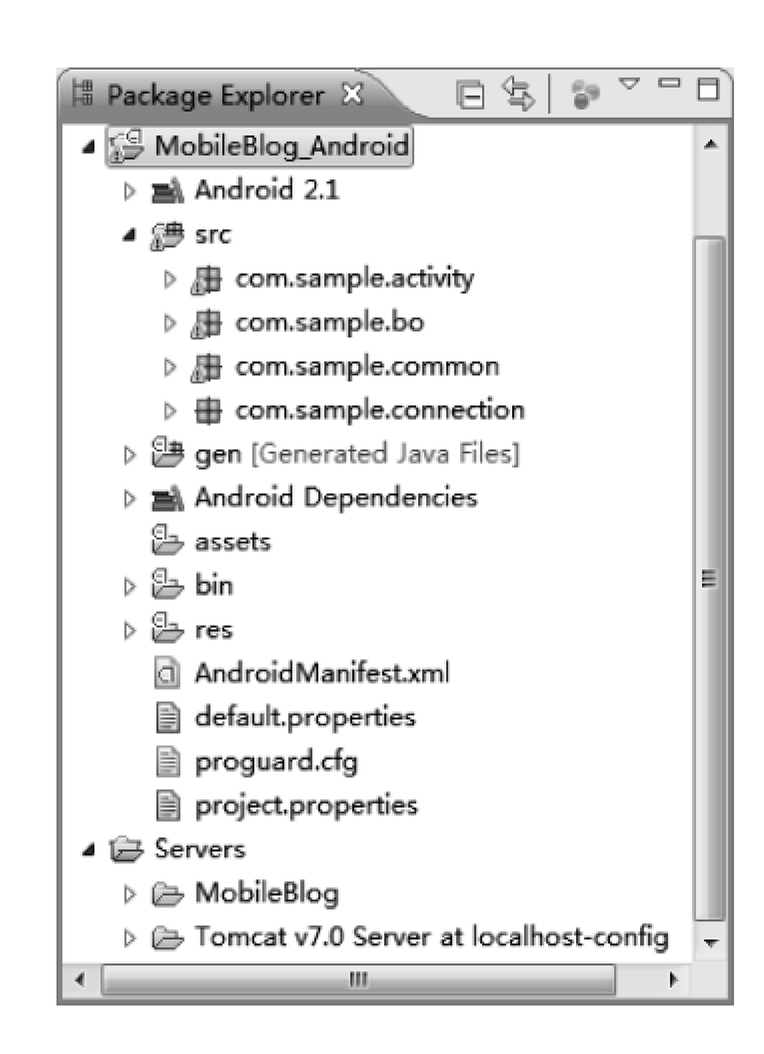


图 12-24　Eclipse 中的项目目录

展开 res 目录，可看到本项目中使用了若干图片资源和预定义字符资源，13 个布局文件，如图 12-25 所示。展开 src 目录，可看到本项目中的逻辑代码分别使用了 4 个包，如图 12-26 所示。其中，在 com. sample. activity 包中是项目的全部 Activity 的类代码定义，在 com. sample. bo 包中是项目中与业务有关的对象存取方法类定义，在 com. sample. common 包中是项目里面公共的常量和工具类定义，com. sample. connection 包中是项目与网络进行通信连接的类定义。

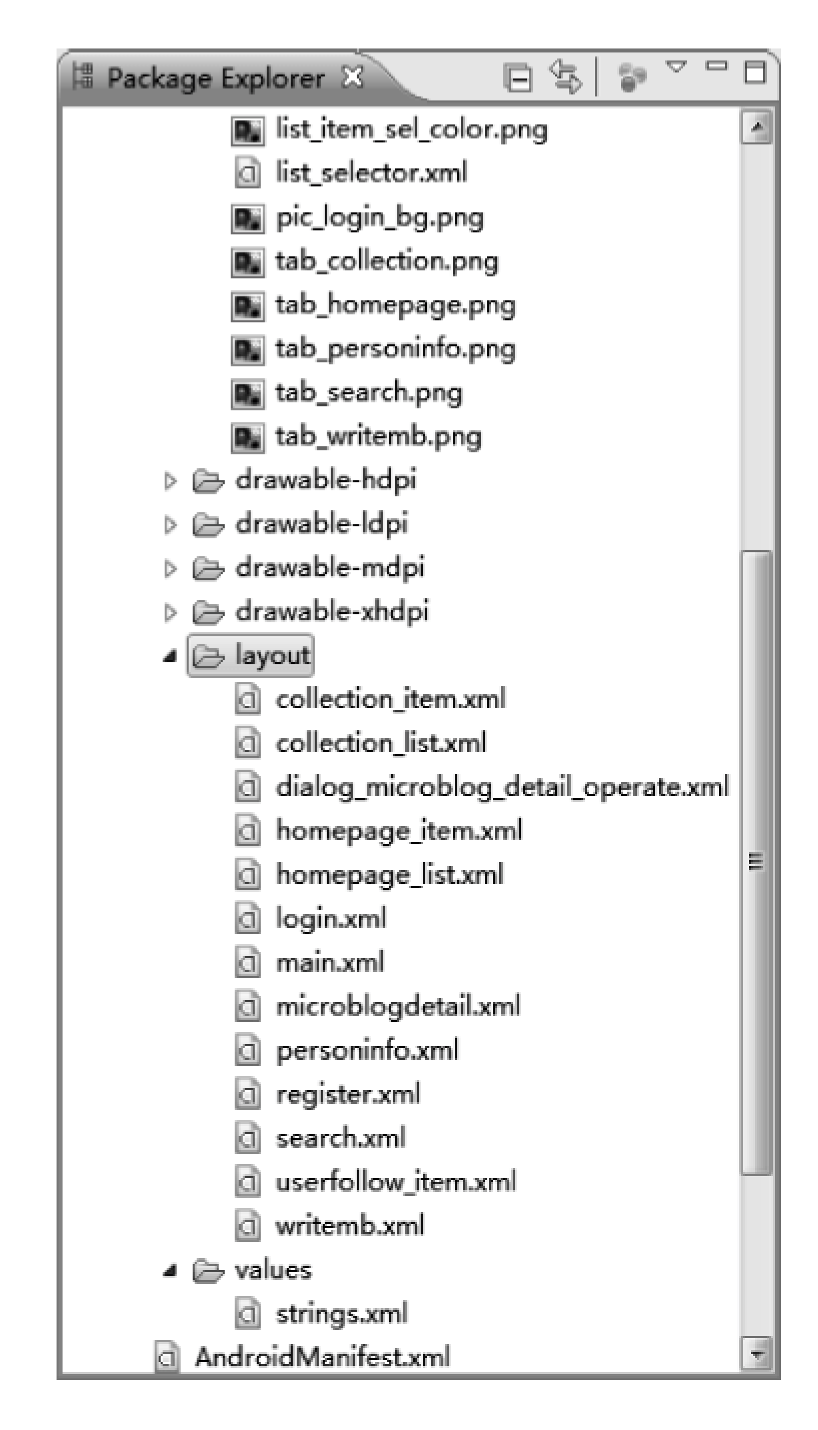


图 12-25　项目的资源目录内容

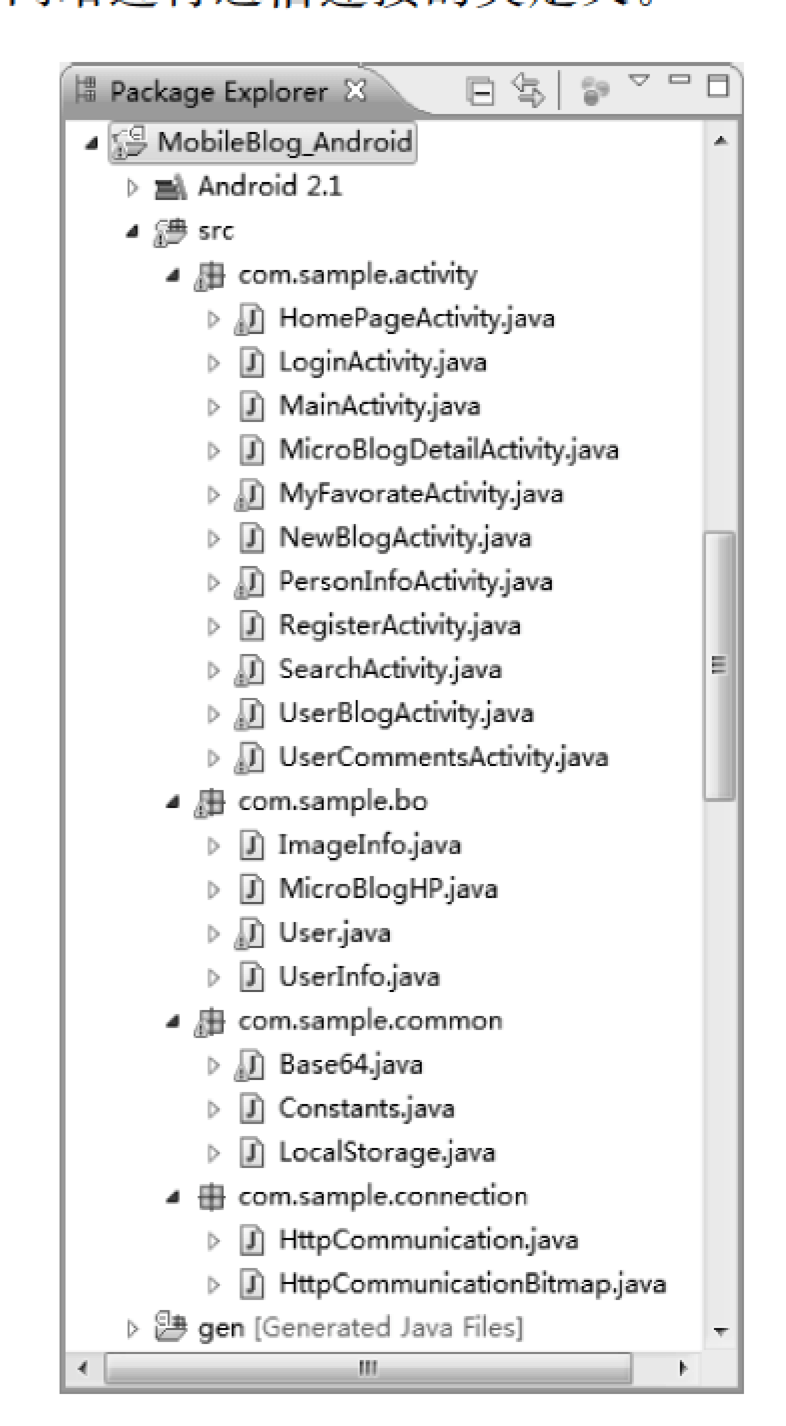


图 12-26　项目的 src 目录内容

12.3.3 功能实现解析

1. 各功能模块的实现类简介

(1) 登录模块由 LoginActivity 类实现。该 Activity 是手机微博运行后首先被启动的 Activity。

(2) 注册模块由 RegisterActivity 类实现。该 Activity 从 LoginActivity 中启动。

(3) 个人中心模块由 MainActivity 类实现。该 Activity 从 LoginActivity 中启动,它继承自 TabActiity 类,将手机微博的各主要功能以选项卡的形式显示在屏幕上,并对选项卡进行逻辑控制。

(4) 手机客户端首页模块由 HomePageActivity 和 MicroBlogDetailActivity 类实现,其中,HomePageActivity 类显示若干博客的列表,并可以对列表进行分页,这些博客包括当前用户发布或转发的所有博客及当前用户所关注的用户发布或转发的所有博客。单击任一博客列表项即可打开该博客详细信息。MicroBlogDetailActivity 类显示博客的详细信息,并可对该博客进行转发、评论和收藏。

(5) 发微博模块由 NewBlogActivity 类实现。在该 Activity 中用户可以写博客,本客户端允许用户写 150 个字符的博客内容。在编辑区的下方实时给出还能输入的字符个数,以提示用户注意。

(6) 个人信息模块由 PersonInfoActivity、UserBlogActivity 和 UserCommentsActivity 类实现。其中,PersonInfoActivity 类显示当前用户的主要信息,包括昵称、年龄、最后的登录时间和发表的博客数、关注的用户数和被其他用户关注即粉丝数信息。单击博客数,可进入 UserBlogActivity;单击关注数和粉丝数,可进入 UserCommentsActivity。

(7) 收藏管理模块由 MyFavorateActivity 类实现。该 Activity 以列表的方式显示被当前用户所关注的所有用户列表,当长按某项关注项时,会出现删除此关注项的对话框。

(8) 查找模块由 SearchActivity 类实现。在该 Activity 的编辑框内输入要查找的用户名或用户名的前几个字串,单击"查找"图标,即可在网络服务器中查找到所有以输入字符开头的用户名。当在某条用户上长按时,会出现关注该用户的对话框。

2. 功能模块与服务器的通信实现

本手机客户端程序与服务器数据库的数据通信是通过接口,采用 HTTP 请求获取网络服务器资源。通信接口以 UTF-8 进行编码,使用 post 请求方式,数据返回值采用普通字符串或 JSON 格式进行封装。

问一下:

什么是 JSON 格式?

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于 JavaScript (Standard ECMA-262 3rd Edition-December 1999)的一个子集,采用完全独立于语言的文本格式,易于人阅读和编写,同时也易于机器解析和生成。这些特性使得 JSON 成为理想的数据交换语言。

1）与服务器端通信接口协议

在本实例中，调用接口方法是：

```
String HttpCommunication.sendPostHttpRequest(String url, List<NameValuePair> params);
```

其中，输入参数1，String url 是接口的 URL；输入参数 2，List params 是 post 到服务器的参数列表。这些参数是 key/value 的值对列表。

返回值：普通字符串或者 JSON 格式的字符串。

表 12-9～表 12-23 给出手机客户端各应用模块的接口协议说明。

表 12-9　登录模块接口

接 口 说 明	**登 录 接 口**			
URL 地址	http://ip:port/MobileBlog/ClientUserAction!login.action			
参数列表				
参数名	类型	描述	值条件	备注
user.name	String	用户名	必填	测试账号：user1
user.password	String	用户密码	必填	测试密码：user
返回参数				
参数名	类型	描述		
result	String	返回的字符串		
返回数据格式				
字段名	说明			备注
success_rec_no	返回一个字符串，字符串由两部分组成："success_"和 rec_no。rec_no 是指该用户所在的数据库的 u_id 字段值			例如：一个用户的记录在 tb_user 表里，u_id 字段的值是 3，则返回的值为"success_3"

表 12-10　注册并判断是否有重复用户模块接口

接 口 说 明	**注册时，判断是否有同名的用户**			
URL 地址	http://ip:port/MobileBlog/ClientUserAction!isNameConflict.action			
参数列表				
参数名	类型	描述	值条件	备注
user.name	String	用户名	必填	用户注册时填入的用户名
返回参数				
参数名	类型	描述		
result	String	返回的字符串		
返回数据格式				
字段名	说明			备注
true	说明系统中有同名的用户			
false	说明系统中没有同名的用户			

表 12-11　上传照片模块接口

接口说明	用户通过客户端上传图片			
URL 地址	http://ip:port/MobileBlog/UploadImageAction!upload.action			
参数列表				
参数名	类型	描述	值条件	备注
filename	String	文件名	必填	文件的全路径和文件名
filebody	String	文件的内容	必填	应该用 Base64 进行编码
返回参数				
参数名	类型	描述		
result	String	返回的字符串		
返回数据格式				
字段名	说明			备注
success	文件上传成功			
error	文件上传失败			

表 12-12　用户注册模块接口

接口说明	新用户注册的接口			
接口地址	http://ip:port/MobileBlog/ClientUserAction!register.action			
输入参数				
参数名	类型	描述	值条件	备注
user.name	String	注册的用户名	必填	
user.password	String	用户的密码	必填	
user.trueName	String	用户的真实姓名	必填	
user.age	String	用户年龄	必填	
user.picture	String	上传图片的名称	必填	
返回参数				
参数名	类型	描述		
result	JSON 数据	见返回参数说明。如返回数据错误，则根据错误代码构建 JSON 数据		
返回 JSON 数据格式				
字段名	说明			备注
status	操作的状态			
showNote	操作的提示信息			
返回 JSON 数据格式举例				
{"status":"1","showNote":"用户注册成功"}				
返回错误码				
错误码	说明			
－1	用户已存在			
0	注册失败			
返回错误码 JSON 格式举例				
{"status":"－1","showNote":"用户已经存在"} {"status":"0","showNote":"注册失败"}				

表 12-13 获取微博列表模块接口

<table>
<tr><th>接 口 说 明</th><th colspan="4">获取微博列表接口</th></tr>
<tr><td>接口地址</td><td colspan="4">http://ip:port/MobileBlog/ClientMicroBlogAction!getMicroBlogInfo.action</td></tr>
<tr><td colspan="5">输入参数</td></tr>
<tr><td>参数名</td><td>类型</td><td>描述</td><td>值条件</td><td>备注</td></tr>
<tr><td>pageNow</td><td>String</td><td>当前列表页码</td><td>必填</td><td>用来做翻页操作,记录当前页码</td></tr>
<tr><td>userId</td><td>String</td><td>用户 id</td><td>必填</td><td>表示获取的是该用户的微博</td></tr>
<tr><td colspan="5">返回参数</td></tr>
<tr><td>参数名</td><td>类型</td><td colspan="3">描述</td></tr>
<tr><td>JSON 数据</td><td>JSON 数据</td><td colspan="3">见返回参数说明。如返回数据错误,则根据错误代码构建 JSON 数据</td></tr>
<tr><td colspan="5">返回数据格式</td></tr>
<tr><td>字段名</td><td colspan="3">说明</td><td>备注</td></tr>
<tr><td>pageCount</td><td colspan="3">记录分页后的总页数</td><td></td></tr>
<tr><td>results</td><td colspan="3">返回的博客列表</td><td>参加下面“results 说明”部分</td></tr>
<tr><td colspan="5">results 说明</td></tr>
<tr><td>字段名</td><td colspan="3">说明</td><td>备注</td></tr>
<tr><td>content</td><td colspan="3">博客内容</td><td>字符串类型</td></tr>
<tr><td>microBlogId</td><td colspan="3">博客的 ID</td><td>整数</td></tr>
<tr><td>time</td><td colspan="3">博客发表的日期</td><td>年-月-日格式</td></tr>
<tr><td>userId</td><td colspan="3">用户 ID</td><td>发表该博客的用户 ID</td></tr>
<tr><td>userName</td><td colspan="3">用户名</td><td>发表该博客的用户名称</td></tr>
<tr><td>userPhoto</td><td colspan="3">用户头像图片的文件名</td><td>例如：用默认头像,该值为"user_headphoto.png"</td></tr>
<tr><td colspan="5">返回 JSON 数据格式举例</td></tr>
<tr><td colspan="5">{"pageCount":"178","results":[{"content":"jhgg","microBlogId":1205,"time":"2012-07-13","userId":67,"userName":"as","userPhoto":"user_headphoto.png"},{"content":"jjhgcv ","microBlogId":1206,"time":"2012-07-13","userId":67,"userName":"as","userPhoto":"user_headphoto.png"}]}</td></tr>
</table>

表 12-14 转发微博模块接口

<table>
<tr><th>接 口 说 明</th><th colspan="5">转发微博接口</th></tr>
<tr><td>接口地址</td><td colspan="5">http://ip:port/MobileBlog/ClientMicroBlogAction!microBlogTransmit.action</td></tr>
<tr><td colspan="6">输入参数</td></tr>
<tr><td>参数名</td><td>类型</td><td colspan="2">描述</td><td>值条件</td><td>备注</td></tr>
<tr><td>transmit.microBlog.id</td><td>String</td><td colspan="2">被转发微博 ID</td><td>必填</td><td></td></tr>
<tr><td>transmit.user.id</td><td>String</td><td colspan="2">被转发微博用户 ID</td><td>必填</td><td></td></tr>
<tr><td>transmit.content</td><td>String</td><td colspan="2">被转发微博的内容</td><td>必填</td><td></td></tr>
<tr><td colspan="6">返回参数</td></tr>
<tr><td>参数名</td><td>类型</td><td colspan="4">描述</td></tr>
<tr><td>result</td><td>String</td><td colspan="4">见返回参数说明</td></tr>
<tr><td colspan="6">返回数据格式</td></tr>
<tr><td>字段名</td><td>说明</td><td>字段类型</td><td colspan="3">备注</td></tr>
<tr><td>success</td><td>转发成功</td><td>String</td><td colspan="3"></td></tr>
</table>

表 12-15 发表微博评论模块接口

接 口 说 明	发表微博评论接口			
接口地址	http://ip:port/MobileBlog/ClientMicroBlogAction!microBlogComment.action			
输入参数				
参数名	类型	描述	值条件	备注
comment.microBlog.id	String	被评论微博 ID	必填	
comment.user.id	String	被评论微博用户 ID	必填	
comment.content	String	被评论微博的内容	必填	
返回参数				
参数名	类型	描述		
result	String	见返回参数说明		
返回数据格式				
字段名	说明	字段类型	备注	
success	评论成功	String		

表 12-16 收藏微博模块接口

接 口 说 明	收藏微博接口			
接口地址	http://ip:port/MobileBlog/ClientMicroBlogAction!microBlogCollection.action			
输入参数				
参数名	类型	描述	值条件	备注
collection.microBlog.id	String	被收藏微博 ID	必填	
collection.user.id	String	被收藏微博用户 ID	必填	
返回参数				
参数名	类型	描述		
result	String	见返回参数说明		
返回数据格式				
字段名	说明	字段类型	备注	
success	收藏成功	String		

表 12-17 发表新微博模块接口

接 口 说 明	发表新微博接口			
接口地址	http://ip:port/MobileBlog/ClientMicroBlogAction!saveMicroBlog.action			
输入参数				
参数名	类型	描述	值条件	备注
microBlog.content	String	微博内容	必填	
userId	String	发表微博的用户 ID	必填	
返回参数				
参数名	类型	描述		
result	String	见返回参数说明		
返回数据格式				
字段名	说明	字段类型	备注	
success	发表微博成功	String		

表 12-18 显示用户资料模块接口

接 口 说 明	从服务器端获取某个用户的资料			
接口地址	http://ip:port/MobileBlog /ClientMicroBlogAction!getUserInfo.action			
输入参数				
参数名	类型	描述	值条件	备注
userID	String	用户 ID	必填	
返回参数				
参数名	类型	描述		
result	String	见返回参数说明		
Result 包含的数据说明				
字段名	说明			备注
fansCount	粉丝数			整型
idolCount	收听人数			整型
lastLoginTime	上次登录日期			字符串：YYYY-MM-DD
mbCount	以发表微博数			整型
userAge	用户年龄			整型
userId	用户编号			整型
userName	用户姓名			字符串
userPicture	用户头像的 URL			字符串
返回 JSON 数据格式说明				
{"results":[{"fansCount":7,"idolCount":13,"lastLoginTime":"2012-08-10","mbCount":9,"userAge":"28","userId":1,"userName":"user1","userPicture":"20110418221357.jpg"}]}				

表 12-19 关注微博模块接口

接 口 说 明	关注新微博接口			
接口地址	http://ip:port/MobileBlog/ClientMicroBlogAction!saveUserIdol.action			
输入参数				
参数名	类型	描述	值条件	备注
follow.idol.id	String	被关注用户 ID	必填	
follow.fans.id	String	关注者用户 ID	必填	
返回参数				
参数名	类型	描述		
result	String	见返回参数说明		
返回数据格式				
字段名	说明	字段类型	备注	
success	关注成功	String		

表 12-20　取消关注微博模块接口

接 口 说 明	取消关注新微博接口			
接口地址	http://ip:port/MobileBlog/ClientMicroBlogAction!deleteUserIdol. action			
输入参数				
参数名	类型	描述	值条件	备注
follow. idol. id	String	被关注用户 ID	必填	
follow. fans. id	String	关注者用户 ID	必填	
返回参数				
参数名	类型	描述		
result	String	见返回参数说明		
返回数据格式				
字段名	说明	字段类型	备注	
success	取消关注成功	String		

表 12-21　搜索用户模块接口

接 口 说 明	搜索用户接口			
接口地址	http://ip:port/MobileBlog/ClientMicroBlogAction!getMicroBlogInfo. action			
输入参数				
参数名	类型	描述	值条件	备注
pageNow	String	当前列表页码	必填	用来做翻页操作,记录当前页码
search	String	用于搜索的关键字	必填	
返回参数				
参数名	类型	描述		
JSON 数据	JSON 数据	见返回参数说明。如返回数据错误,则根据错误代码构建 JSON 数据		
返回数据格式				
字段名	说明	备注		
pageCount	返回记录的分页数目			
results	返回符合条件的用户信息列表	详见下面 results 列表数据说明		
results 说明				
字段名	说明	备注		
fansCount	听众数量	整数		
idolCount	收听的用户数	整数		
lastLoginTime	最后一次登录的日期	年-月-日格式		
mbCount	发表博客的数量	整数		
userAge	用户年龄	整数		
userId	用户 ID			
userName	用户名称			
userPicture	用户的头像文件名	例如:用默认头像,该值为“user_headphoto. png”		
返回 JSON 数据格式举例				
{"pageCount":"5","results":[{"fansCount":4,"idolCount":6,"lastLoginTime":"2011-05-28","mbCount":13,"userAge":"25","userId":2,"userName":"user2","userPicture":"20110418221453. jpg"},{"fansCount":4,"idolCount":7,"lastLoginTime":"2011-03-27","mbCount":33,"userAge":"24","userId":3,"userName":"user3","userPicture":"20110418221357. jpg"}]}				

表 12-22　获取收藏微博列表模块接口

接 口 说 明	获取收藏微博列表接口			
接口地址	http://ip:port/MobileBlog/ClientMicroBlogAction!getCollectionInfo.action			
输入参数				
参数名	类型	描述	值条件	备注
pageNow	String	当前列表页码	必填	用来做翻页操作，记录当前页码
userId	String	用户 id	必填	表示获取的是该用户的微博收藏
返回参数				
参数名	类型	描述		
JSON 数据	JSON 数据	见返回参数说明。如返回数据错误，则根据错误代码构建 JSON 数据		
返回数据格式				
字段名	说明			备注
pageCount	记录分页后的总页数			
results	返回的博客列表			参加下面“results 说明”部分
results 说明				
字段名	说明			备注
content	博客内容			字符串类型
microBlogId	博客的 ID			整数
time	博客发表的日期			年-月-日格式
userId	用户 ID			发表该博客的用户 ID
userName	用户名			发表该博客的用户名称
userPhoto	用户头像图片的文件名			例如：用默认头像，该值为"user_headphoto.png"
返回 JSON 数据格式举例				
{"pageCount":"178","results":[{"content":"jhgg","microBlogId":1205,"time":"2012-07-13","userId":67,"userName":"as","userPhoto":"user_headphoto.png"},{"content":"jjhgcv ","microBlogId":1206,"time":"2012-07-13","userId":67,"userName":"as","userPhoto":"user_headphoto.png"}]}				

表 12-23　删除收藏模块接口

接 口 说 明	删除收藏接口			
接口地址	http://ip:port/MobileBlog/ClientMicroBlogAction!deleteCollection.action			
输入参数				
参数名	类型	描述	值条件	备注
collection.microBlog.id	String	被收藏的微博 ID	必填	
collection.user.id	String	收藏者用户 ID	必填	
返回参数				
参数名	类型	描述		
result	String	见返回参数说明		
返回数据格式				
字段名	说明		字段类型	备注
success	删除收藏成功		String	

2）客户端数据通信的代码

在本实例中，手机客户端与网络服务器的数据通信方法 sendPostHttpRequest()定义在本项目的 com.sample.connection 包下 HttpCommunication 类中，其代码如下。

```
package com.sample.connection;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.List;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;

import android.util.Log;

import com.sample.common.Constants;

public class HttpCommunication {

  private static HttpClient httpClient = null;

  public static HttpClient getHttpClient() {
        if (httpClient == null) {
            httpClient = new DefaultHttpClient();
        }
        return httpClient;
    }

    public static String sendPostHttpRequest(String url,
            List<NameValuePair> params) {
        //设置 HttpPost 连接对象(HttpPost 即为 HttpRequest)
        HttpPost httpPost = new HttpPost(Constants.SERVERADDRESS + url);
        try {
            //设置字符集为 utf-8
            HttpEntity httpEntity = new UrlEncodedFormEntity(params, "utf-8");
            //使用 HttpPost 类的 setEntity()方法设置请求参数
            httpPost.setEntity(httpEntity);
            //使用 DefaultHttpClient 为 HttpClient
            HttpClient httpClient = getHttpClient();
            //取得 HttpResponse 对象
            HttpResponse httpResponse = httpClient.execute(httpPost);
            //HttpStatus.SC_OK 表示连接成功
            if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
                //取得返回的字符串
                String result = "";
                if (httpResponse.getEntity() != null) {
                    InputStream is = httpResponse.getEntity().getContent();
                    result = convertStreamToString(is);      //将输入流转换为字符串
                    return result.trim();
                }
```

```
            } else {
                System.out.println("fail the request");
            }

        } catch (Exception e) {
            //TODO: handle exception
            e.printStackTrace();
        }
        return null;
    }

    public static String convertStreamToString(InputStream ism) {
        BufferedReader reader = null;
        try {
            reader = new BufferedReader(new InputStreamReader(ism, "UTF-8"),
                                                        //防止模拟器上的乱码
                    512 * 1024);
        } catch (UnsupportedEncodingException e1) {

            e1.printStackTrace();
        }
        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
        } catch (IOException e) {
            Log.e("DataProvier convertStreamToString", e.getLocalizedMessage(),
                    e);
        } finally {
            try {
                ism.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return sb.toString();
    }
}
```

上述类代码主要定义了 sendPostHttpRequest()方法。该方法完成向服务器端传送 UTF-8 参数，同时也接收从服务器传过来的数据流，并通过 convertStreamToString()方法将数据流转换为字符串形式。

在 HomePageActivity、MyFavorateActivity、SearchActivity、UserBlogActivity、UserBlogActivity 等多个类中，定义了方法 shareJSON()。该方法主要用于实现把包含 JSON 数据结构的字符串解析出来，保存成序列化的对象。其代码片段如下。

```
……
private void shareJSON() {
        String url = "/ClientMicroBlogAction!getMicroBlogInfo.action";
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        //表单参数
```

```
            params.add(new BasicNameValuePair("pageNow", pageNow + ""));
            params.add(new BasicNameValuePair("userId", LocalStorage.getString(
                    HomePageActivity.this, "userId")));

            Log.i(TAG,
                    "getUserId : "
                            + LocalStorage.getString(HomePageActivity.this,
                                    "userId"));
            //返回 JSON 字符串,并解析
            json = HttpCommunication.sendPostHttpRequest(url, params);

            try {
                //获取根节点
                JSONObject root = new JSONObject(json.toString());
                pageCount = root.getInt("pageCount");
                Log.i(TAG, "pageCount >>>>>     " + pageCount);
                JSONArray items = root.getJSONArray("results");
                for (int i = 0; i < items.length(); i++) {
                    MicroBlogHP share = new MicroBlogHP();
                    share.setMicroblogId(Integer.parseInt(items.getJSONObject(i)
                            .getString("microBlogId")));
                    share.setUserId(Integer.parseInt(items.getJSONObject(i)
                            .getString("userId")));
                    share.setUserName(items.getJSONObject(i).getString("userName"));
                    share.setUserPicture(Constants.SERVERADDRESS + "/headPhoto/"
                            + items.getJSONObject(i).getString("userPhoto"));
                    share.setContent(items.getJSONObject(i).getString("content"));
                    share.setTime(items.getJSONObject(i).getString("time"));
                    list.add(share);
                }

            } catch (JSONException e) {

                e.printStackTrace();
            }

        }
……
```

shareJSON()方法向网络服务器中传入两个参数 url 和 params,其中参数 url 的值是提供处理该数据通信的接口地址,例如“url = "/ClientMicroBlogAction! getMicroBlogInfo.action";”表示这个 HTTP 请求由服务器的 ClientMicroBlogAction 类里面的 getMicroBlogInfo()方法来负责处理。参数 params 的值是传入服务器的键值对,上述代码中的 params 包含博客列表的当前页码和用户 ID 号。

3. 部分代码解析

1) 获取头像

本实例在用户注册的 RegisterActivity 活动中获取头像的方式有两种:一是调用 Android 自带的拍照应用程序进行拍照获取,使用 Android 自带的拍照应用时使用的 Intent 参数是 MediaStore.ACTION_IMAGE_CAPTURE;二是从 SD 卡的相册中获取头像,SD 卡的 DCIM/Camera 目录是相册的存储位置。相关的代码如下。

```
……
/**
  * 拍照获取图片
  *
  */
  protected void doTakePhoto() {
      try {
          //Launch camera to take photo for selected contact
          PHOTO_DIR.mkdirs();                                  //创建照片的存储目录
          mCurrentPhotoFile = new File(PHOTO_DIR, getPhotoFileName());//给新照的照片
                                                                  //文件命名
          final Intent intent = getTakePickIntent(mCurrentPhotoFile);
          startActivityForResult(intent, CAMERA_WITH_DATA);
      } catch (ActivityNotFoundException e) {
          Toast.makeText(this, R.string.photoPickerNotFoundText,
                  Toast.LENGTH_LONG).show();
      }
  }

  public static Intent getTakePickIntent(File f) {
      Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE, null);
      intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(f));
      return intent;
  }

  /**
   * 用当前时间给取得的图片命名
   *
   */
  private String getPhotoFileName() {
      Date date = new Date(System.currentTimeMillis());
      SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMddhhmmss");
      return dateFormat.format(date) + ".png";
  }

  //请求 Gallery 程序,从相册中去获取
  protected void doPickPhotoFromGallery() {
      try {
          //Launch picker to choose photo for selected contact
          final Intent intent = getPhotoPickIntent();
          startActivityForResult(intent, PHOTO_PICKED_WITH_DATA);
      } catch (ActivityNotFoundException e) {
          Toast.makeText(this, R.string.photoPickerNotFoundText1,
                  Toast.LENGTH_LONG).show();
      }
  }

  //封装请求 Gallery 的 intent
  public static Intent getPhotoPickIntent() {
      Intent intent = new Intent(Intent.ACTION_GET_CONTENT, null);
      intent.setType("image/*");
      intent.putExtra("crop", "true");
      intent.putExtra("aspectX", 1);
      intent.putExtra("aspectY", 1);
      intent.putExtra("outputX", 80);
```

```
        intent.putExtra("outputY", 80);
        intent.putExtra("return-data", true);
        return intent;
    }

    //因为调用了 Camera 和 Gallery 所以要判断它们各自的返回情况,它们启动时是使用
    //startActivityForResult()方法
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode != RESULT_OK)
            return;
        switch (requestCode) {
        case PHOTO_PICKED_WITH_DATA: {//调用 Gallery 返回的
            final Bitmap photo = data.getParcelableExtra("data");
            //下面就是显示照片了
            System.out.println(photo);
            //缓存用户选择的图片
            img = getBitmapByte(photo);
            Log.v("RegisterActivity", "new photo set!");
            //mEditor.setPhotoBitmap(photo);
            iv_headPhoto.setImageBitmap(photo);
            System.out.println("set new photo");
            photoName = getPhotoFileName();
            break;
        }
        case CAMERA_WITH_DATA: {//照相机程序返回的,再次调用图片剪辑程序去修剪图片
            doCropPhoto(mCurrentPhotoFile);
            break;
        }
        }
    }

    //将字节数组转换为 bitmap
    public Bitmap getBitmapFromByte(byte[] temp) {
        if (temp != null) {
            Bitmap bitmap = BitmapFactory.decodeByteArray(temp, 0, temp.length);
            return bitmap;
        } else {
            return null;
        }
    }

    //将头像转换成 byte[]以便能将图片上传到服务器
    public byte[] getBitmapByte(Bitmap bitmap) {
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, out);
        return out.toByteArray();
    }

    protected void doCropPhoto(File f) {
        try {
            final Intent intent = getCropImageIntent(Uri.fromFile(f));
            startActivityForResult(intent, PHOTO_PICKED_WITH_DATA);
        } catch (Exception e) {
            Toast.makeText(this, R.string.photoPickerNotFoundText,
                    Toast.LENGTH_LONG).show();
```

```
            }
        }

        /**
         * Constructs an intent for image cropping. 调用图片剪辑程序
         */
        public static Intent getCropImageIntent(Uri photoUri) {
            Intent intent = new Intent("com.android.camera.action.CROP");
            intent.setDataAndType(photoUri, "image/*");
            intent.putExtra("crop", "true");
            intent.putExtra("aspectX", 1);
            intent.putExtra("aspectY", 1);
            intent.putExtra("outputX", 80);
            intent.putExtra("outputY", 80);
            intent.putExtra("return-data", true);
            return intent;
        }
    ……
```

代码中，doTakePhoto()、getTakePickIntent()方法是定义启用 Android 自带的拍照应用；getPhotoFileName()方法为图像文件命名；doPickPhotoFromGallery()、getPhotoPickIntent()方法是定义从 SD 卡的相册中取头像操作；onActivityResult()方法是处理调用拍照(Camera)和相册(Gallery)的返回情况；getBitmapFromByte()、getBitmapByte()、doCropPhoto()、getCropImageIntent()这些方法是对图像进行存储格式转换或剪辑等加工处理的方法。

2) 计算显示页

在 HomePageActivity 中显示博客列表时，当博客数量较多时，一屏显示不下，系统自动对博客列表进行分页。如果向下滑动，可以看到第 2 页、第 3 页等列表项。相关代码如下。

```
……
        listView.setOnScrollListener(new OnScrollListener() {

            @Override
            public void onScrollStateChanged(AbsListView view, int scrollState) {
                //TODO Auto-generated method stub
                Log.i(TAG, "onScrollStateChanged ");
            }

             @Override
             public void onScroll(AbsListView view, int firstVisibleItem,
                     int visibleItemCount, int totalItemCount) {
                 //TODO Auto-generated method stub
                 Log.i(TAG, "Scroll>>>first: " + firstVisibleItem
                         + ", visible: " + visibleItemCount + ", total: "
                         + totalItemCount);
                 //是否还有可加载页
                 if (pageNow < pageCount) {
                     //判断是否滑动到底部
                     if (firstVisibleItem + visibleItemCount == totalItemCount) {
                         pageNow += 1;
                         //加载数据
                         shareJSON();
                         //重绘 listview
                         adapter.notifyDataSetChanged();
```

```
                        //显示最新加载的最上面一条
                        listView.setSelection(firstVisibleItem
                                + visibleItemCount - 2);

                        Toast.makeText(HomePageActivity.this,
                                "第" + pageNow + "页", Toast.LENGTH_LONG).show();
                    }
                } else {
                    //最后一页去除加载进度条
                    listView.removeFooterView(loadingLayout);
                }
            }
        });

    }
    ……
```

在 listView.setOnScrollListener()监听接口中来定义相关代码。listView 列表项是按页显示博客项的，具体处理分页的代码在服务器控制程序中定义，在手机客户端只是通过接口进行加载数据即可。

3）计算编辑区剩余字数

在 NewBlogActivity 中进行编辑博客时，可以对编辑过程实施监听，计算出剩余的输入字符个数。相关代码如下。

```
……
@Override
protected void onCreate(Bundle savedInstanceState) {
    //TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.writemb);

    et_mb_content = (EditText) findViewById(R.id.tab_wmb_Content);
    ……
     et_mb_content.addTextChangedListener(new TextWatcher() {
        private CharSequence temp;

        @Override
        public void onTextChanged(CharSequence s, int start, int before,
                int count) {
            //TODO Auto-generated method stub

        }

        @Override
        public void beforeTextChanged(CharSequence s, int start, int count,
                int after) {
            //TODO Auto-generated method stub
            temp = s;
        }

        @Override
        public void afterTextChanged(Editable s) {
            //TODO Auto-generated method stub
            int size = 150 - temp.length();
```

```
                tv_mb_contentSize.setText(size + "");
            }
        });
……
}
```

代码中，对 EditText 对象添加监听器，其监听接口为 addTextChangedListener()。在其中，onTextChanged()、beforeTextChanged()、afterTextChanged ()分别是修改中、修改前、修改后的回调事件方法。

12.4 手机客户端的测试运行

在开发完成应用程序后都需要对其进行测试运行，通常是先在模拟器中进行测试，待测试通过之后再签名打包，发布到手机真机中运行。在模拟器中运行手机微博程序时，需要保证 Tomcat 服务器是开启状态，并且检查代码中的 IP 地址与本机 IP 一致等准备工作。

1. 运行前的准备工作

(1) 查看本机 IP 地址。

在命令行状态下输入 ipconfig 命令，即可看到本机的 IP 地址等信息，如图 12-27 所示。



图 12-27 查看本机的 IP 信息

然后检查本应用项目中两个代码文件中的 IP 地址是否与本机 IP 一致：打开 Constants.java 文件，检查 SERVERADDRESS 常量所指的 IP 地址是否与本机 IP 一致，打开 Strings.xml 文件，检查＜string name＝"serverAddress"＞标签中的 IP 地址描述是否与本机 IP 一致。如果不一致，需要修改代码中的 IP 为本机 IP。

(2) 在 Eclipse 中启动 Tomcat 服务器，如图 12-28 所示。

图 12-28 启动 Tomcat 服务器

（3）在 Eclipse 中启动模拟器。本实例使用的 SDK 最低版本为 2.1，所以可以启动 2.1 以上版本（包含 2.1 版本）的模拟器。

2. 运行手机微博应用

在 Eclipse 中选择项目 MobileBlog_Android，然后右击，在弹出菜单中选择 Run As→Android Application，开始运行项目。

1）系统登录

运行 MobileBlog_Android 项目，出现登录界面。首次运行时，“用户名”和“密码”输入框为空，如图 12-29 所示；再次运行此项目，系统记住了上一次登录时使用的用户名和密码，如图 12-30 所示。



图 12-29 首次登录系统



图 12-30 再次登录系统

2）注册新用户

如果需要注册新用户，则单击登录窗口中的“注册”按钮，进入注册窗口，如图 12-31 所示。依次录入注册信息，并且可以通过拍照或从 SD 卡中载入头像来更换原默认头像。注册信息录入完成后单击“注册”按钮，如图 12-32 所示。



图 12-31 进入注册窗口



图 12-32 注册完成

3）进入首页

如果是新注册的用户，进入首页将看不到任何信息，这是因为新注册的用户既没有写一篇博客，也没有关注其他的用户的博客。

如果是老用户，首页将以列表的方式显示本用户所写的博客或转发的博客，以及所关注的用户发送的博客，如图 12-33 所示。如果要查看博客的详情，在某条博客上长按，进入该条博客的详细界面，如图 12-34 所示。



图 12-33 当前用户及关注用户的博客列表



图 12-34 第一条 user1 用户的博客

4）发表博客

在此界面中可以写 150 个字符的微博内容。在写博客的过程中，编辑框下方可以自动实时地计算还能录入的字符数，如图 12-35 所示。

5）个人信息

在个人信息中可以查看当前用户的注册信息，如图 12-36 所示。在图 12-36 中单击“关注(4)”，可以查看详细的关注用户列表，如图 12-37 所示。



图 12-35 写微博界面



图 12-36 个人信息页



图 12-37 查看关注的用户

6）收藏管理

进入收藏页，可以查看当前用户所收藏的用户列表，如图 12-38 所示。单击某条用户，则可执行删除此条收藏信息，如图 12-39 所示。



图 12-38　收藏用户列表



图 12-39　删除一条收藏信息

7）查找用户

在“查找用户”编辑框内输入要查找的用户名，然后单击“查找”图标，可以查找到服务器中所有以输入字符开头的用户名，如图 12-40 所示。如果在某条用户上长按，可出现关注该用户的对话框，如图 12-41 所示。



图 12-40　查找以“user”开头的用户



图 12-41　关注选中用户

12.5　项目打包、签名和发布

在手机微博客户端程序测试运行通过后，接下来就要签名、打包，然后发布使用。在 10.5 节中已经详细地介绍了应用程序的签名打包操作，这里不多作赘述，只针对本实例的签名和打

包进行简单描述。

在 Eclipse 中选择本实例项目 MobileBlog_Android，右击，在弹出菜单中选择 Android Tools→Export Signed Application Packge...，出现如图 12-42 所示的对话框。

单击 Next 按钮，进入签名文件的定义页，完成签名文件名、路径及签名文件密码的定义，如图 12-43 所示。

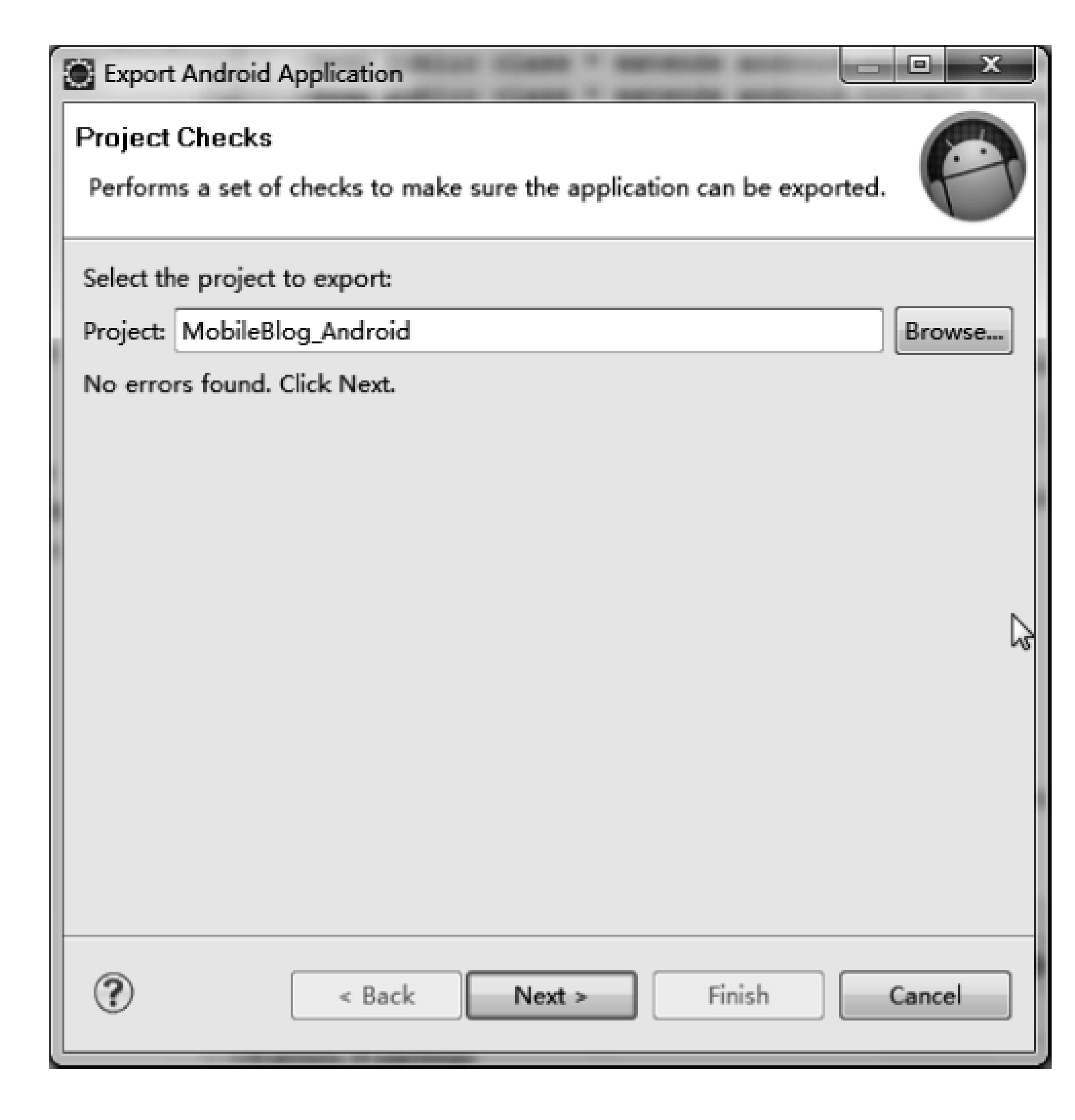


图 12-42　选择签名打包的项目

图 12-43　签名文件定义页

然后单击 Next 按钮，进入生成签名文件信息页，录入相关信息，其中密码和签名文件定义页的一致，如图 12-44 所示。

单击 Next 按钮，进入生成打包文件页，确定打包文件的存放路径和打包文件名，如图 12-45 所示。

图 12-44　生成签名文件信息页

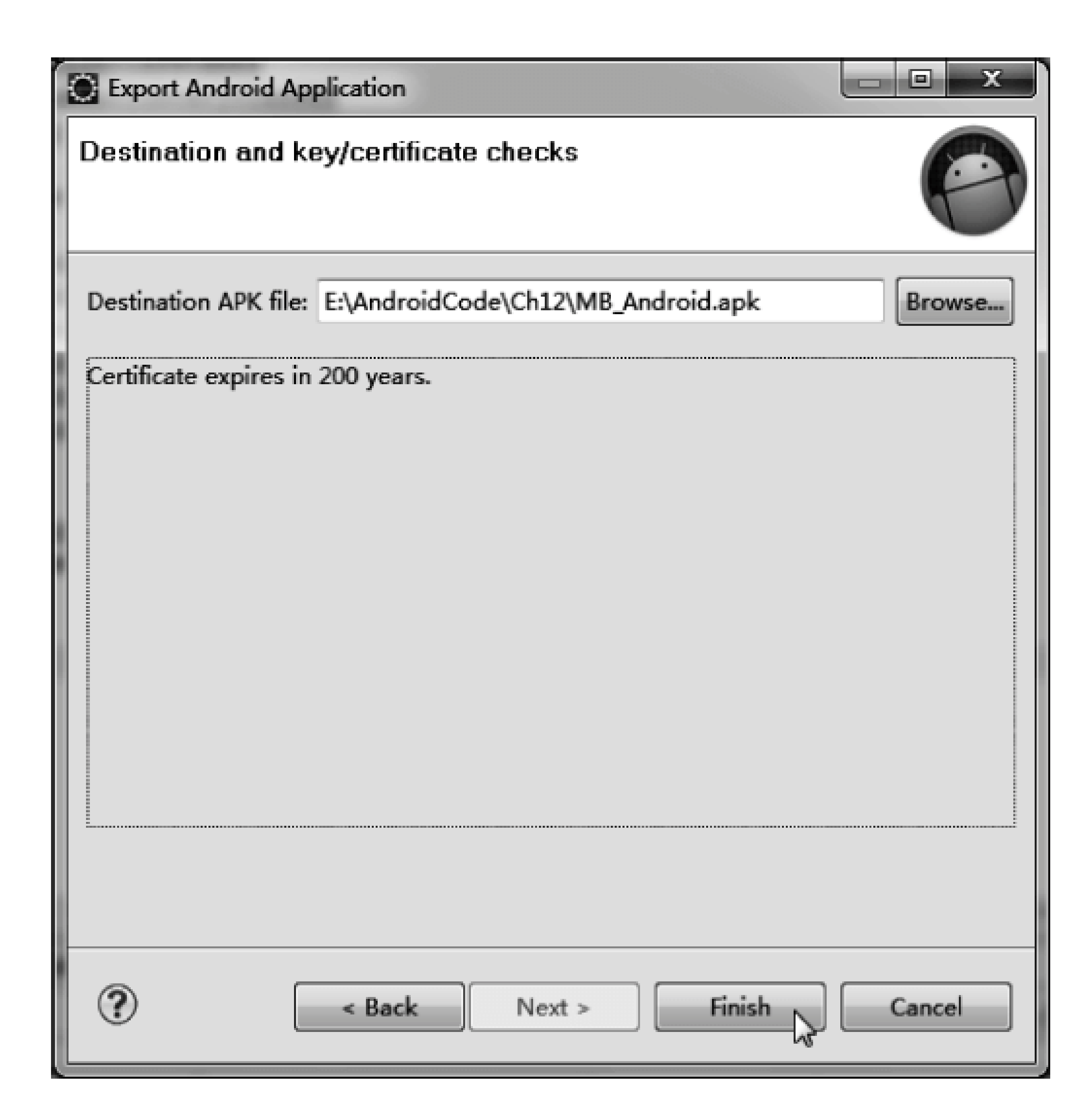


图 12-45　生成打包文件页

然后单击 Finish 按钮，完成本实例的签名和打包，可以在指定的文件夹下（本书指定的文件夹为 E:\AndroidCode\Ch12）看到生成的签名文件和打包文件，如图 12-46 所示。这样得到的打包文件是经过了一个证书文件的加密处理的。现在可以将生成的打包文件上传到网络上，实现手机微博应用程序的发布。



图 12-46　指定文件夹下的内容

小结

本章以开发手机微博客户端应用程序为例，向读者展示了一个实际应用开发的全过程。大多数 Android 应用项目是运行在无线互联网上的应用，需要与网络服务器进行通信，需要与服务器数据库进行数据存取。因此，本章所述主要涉及整个项目的功能、环境设置、数据库及数据库服务器、Android 的手机客户端程序的主要实现及其测试、发布等重点内容。

对于一个实际的项目，Android 手机端的程序设计只是项目中一部分，还需要对服务器端的应用进行编程。本章省略了服务器端应用程序的介绍，只是给出了一个应用程序压缩文件包 MobileBlog. war。如果读者有志在手机应用开发中继续发展，最好学习一下 Java 的 Web 编程课程。

练习

1. 在自己的计算机中安装配置本章介绍的应用项目（包括 Tomcat 服务器，MySQL 数据库环境，数据库表，Web 端应用程序包，Android 手机客户端应用程序等），并运行。理解本书给出的“手机微博客户端”的应用项目范例。

2. 尝试改进与完善本案例的功能。

参 考 文 献

［1］ 靳岩，姚尚朗．Android 开发入门与实战．北京：人民邮电出版社，2009．
［2］ ［美］Ed Burnette．Android 基础教程．张波，高朝勤，杨越等译．北京：人民邮电出版社，2009．
［3］ 吴亚峰，索依娜等．Android 核心技术与实例详解．北京：电子工业出版社，2010．